TURING 图灵程序设计丛书

New Riders

Stunning CSS3 A project-based guide to the latest in CSS

CSS3实用指南

[美] Zoe Mickley Gillenwater 著
屈超 周志超 译

- 设计优美的网站一定要学的CSS3
- 化繁为简，基于实战项目讲解
- 亚马逊网站一致好评

人民邮电出版社
POSTS & TELECOM PRESS

Zoe Mickley Gillenwater

经验丰富的图形和Web设计师、开发人员、顾问。致力于打造尽可能满足各类人群需求的优秀网站，并为最新的浏览器使用创新的CSS技术。她是*Flexible Web Design: Creating Liquid and Elastic Layouts with CSS*和视频培训教程*Web Accessibility Principles*的作者，并已经为Smashing Magazine和Community MX网站撰写了上百篇以Web设计和开发为主题的文章。Zoe目前是Web标准计划小组（WaSP）Adobe任务组的成员之一，还是热门的css-discuss邮件列表的主持人，并经常在全国性会议上讲授CSS、视觉设计、可访问性等主题。

图书在版编目（ＣＩＰ）数据

CSS3实用指南 / (美) 吉伦瓦特
(Gillenwater,Z.M.) 著 ; 屈超, 周志超译. -- 北京 :
人民邮电出版社, 2012.3
（图灵程序设计丛书）
书名原文: Stunning CSS3 : A project-based
Guide to the Latest in CSS
ISBN 978-7-115-27378-9

Ⅰ. ①C… Ⅱ. ①吉… ②屈… ③周… Ⅲ. ①网页制
作工具，CSS－指南 Ⅳ. ①TP393.092-62

中国版本图书馆CIP数据核字(2012)第010442号

内 容 提 要

本书是一本传播很广的关于 CSS3 应用方面的图书。它介绍了如何应用 CSS3 才能比传统方法更加高效、灵活地实现一些常见而优美的网页设计效果。本书通过一系列务实而创新的项目，在实践中为读者讲解 CSS3 应用的前沿技巧。另外本书还整理了每一项 CSS3 属性、选择器和值的使用方法，为读者提供了示例之外的实用性极强的实践指导。

本书适合网站开发人员与网页设计师阅读。

图灵程序设计丛书

CSS3实用指南

◆ 著　[美] Zoe Mickley Gillenwater
译　屈　超　周志超
责任编辑　傅志红
执行编辑　李　盼

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京　印刷

◆ 开本：800×1000　1/16
印张：15.75　彩插 4
字数：372千字　2012年 3 月第 1 版
印数：1－4 000册　2012年 3 月北京第 1 次印刷

著作权合同登记号　图字：01-2011-2429 号

ISBN 978-7-115-27378-9

定价：49.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223

反盗版热线：(010)67171154

版 权 声 明

感谢 Butkus 先生

在北道纳斯格罗夫中学教会我 HTML 和 Photoshop 4。

译　者　序

本书原名为 *Stunning CSS3*，起初我们将其直译为《惊艳的 CSS3》，因为你可以在书中的一些角落找到能够与之呼应的“令人惊艳”一词。现名《CSS3 实用指南》则显得朴素而直接，事实上也更能阐述本书的主旨：除了令大家“一眼惊艳”之外，更重要的还是通过真实的项目实操来全面了解 CSS3 技术。其中最具实用意义的浏览器兼容方案的内容则更是能够帮助你马上将这些新技术运用到实际项目之中——毕竟对于国内的 Web 开发者来说，高居不下的 IE 6 占有率仍是我们眼下无法逃避的现实。

要介绍本书的内容，就不得不提到 CSS（层叠样式表）的发展史。作为 Web 开发中一种可将外观和内容分离的技术手段，CSS 的出现和发展已历经了十几个年头，甚至于今年六月刚刚摘掉“候选”帽子的 CSS 2.1 标准，距离其上一个版本 2.0 也有近十三年之遥了，是时候应该出现一种新的技术标准来驱动更高级的生产方式，以适应互联网日新月异的变化以及对用户体验的极致追求了。因此，早在 2001 年 W3C 就开始了对新的 CSS3 标准的规划工作。时至今日，正如我们所见，在浏览器支持度不断被更新的大趋势下，CSS3 正在为 Web 设计师和开发者开拓一条崭新的道路——它并不满足于对现有功能的扩展与延伸，而是更多地聚焦于对编程模型的简化以及在 Web UI（用户界面）设计理念和方法层面的革新。

在 CSS 2.1 时代，一名优秀的 Web 设计师时常因为需要用到圆角、阴影或渐变等效果而不得不在用户体验和页面性能之间作出取舍，而一名优秀的 Web 开发者则需要对实现这些效果所需的各种浏览器兼容方案烂熟于胸。CSS3 的出现并没有彻底颠覆这一切，但它至少让我们的工作变得更轻松了，同时也让页面拥有了更加丰富的可选效果。但最为重要的是，CSS3 开始采用模块化的管理方式，其规范被划分为若干个相互独立的模块：一方面，较小的模块有利于规范的及时发布与更新，工作组可随时根据现实状况来调整标准；另一方面，由于 CSS3 特性受限于终端的支持程度，因此厂商可以有选择地支持其中的部分模块，以实现对 CSS3 某个子集的支持。这终将有利于 CSS3 的普及与推广，我们也有理由相信这会改变 CSS 支持度从前混乱不堪的局面。截至本书出版之时，CSS3 仍未完整得到所有主流浏览器的支持。

在了解这些情况之后，我们是否应该继续拥抱尚未成熟的 CSS3 技术呢？读完本书，你就能找到答案。本书的作者佐伊・米克莉・吉伦沃特是一名优秀的 Web 设计师，她拥有非常丰富的 Web 设计和开发经验。在本书中，她巧妙地将自身经验与 CSS3 的模块划分相结合，用了七章的篇幅尽可能立体地阐述了 CSS3 里最为重要的技术细节。而从这些或许你之前从未深切关注过的技术细节之中，你不仅可以感受到作者严谨的治学态度，同时也能收获不少对 Web 设计、开发

工作有用的经验总结。与此同时，你还可以从本书中了解到为何我们需要在产品中立即使用 CSS3 技术，以及如何才能规避 CSS3 无法得到所有浏览器支持所带来的诸多问题。作者贯穿全书的开发思想以及对 CSS3 特性的全方位实践，一定能让不同层面的读者都受益匪浅。

致谢

本书的翻译工作由两名译者合作完成，其中周志超翻译了第 1、3、7 章，屈超则负责完成其余的部分。两位译者在这里特别感谢工作团队中的各位同事，没有他们在日常开发中的经验总结，翻译中所遇到的晦涩之处将无据可考。

尤其需要致以诚挚谢意的还有本书编辑团队中的傅志红和李盼两位女士，没有她们在统筹和编辑工作上的辛勤付出，本书绝不可能按期完成。

最后，让我们拥抱未来，拥抱变化，拥抱 CSS3，让本书为你打开一条指向未来 Web 开发世界的通途。

谢谢！

（书中脚注如无特别说明，均为译者注。）

引　　言

CSS3 是最新版的 Web 样式表语言，它不是介绍何创造新颖特效的，而更多地侧重于提供一些神奇的新方法来实现常见而优美的网页设计效果，这些新方法比十年来惯用的传统技术更为高效，而且能够获得可用性更高且灵活的结果。

CSS3 在不断地变化和发展之中，因为支持 CSS 的浏览器在发展，网页设计师对它的认知程度也在不断提升。它可以用来创造一些令人惊艳的酷炫特效，本书将一一予以展现。要是这些特效对于目前的网站不具实用性，那著书意义何在？因此在本书中，我将致力于教给你那些能够切实改善网站体验，并可以迅速应用到日常工作中的 CSS 前沿技巧。

本书并非 CSS3 的百科全书或参考指南，它不会把 CSS2.1 后新增的属性、选择器和值逐个地教给你，而是通过一系列务实与创新的项目来为你讲解 CSS3 里最流行、实用和最常见的那些部分。从第 2 章开始，每一章都会让你从几个实例演练中接触到 CSS3 的新技术，并最终完成一块甚至是一个页面。你可以直接将这些实例改写成自己的项目，或是从中获得启发，创造性地运用你学到的其他新属性、选择器和值。

从某种程度上说，CSS3 不仅是一种新的网页开发方式，它同样是一种新的思考方式。举例来讲，如何应用 `border-image` 这个新增属性就是个难题，因为在从事了多年的网站开发之后，你恐怕还不习惯将图片作为边框设计的一种选项来考虑。因此，我整理了一份清单，里面罗列了如何应用每一项 CSS3 属性、选择器和值的方法，这些方法远不止我们在实例演练中所使用的那一种。我希望能提供足够多的灵感，让你知道如何在项目中实践你的 CSS3 技术，并辅以一些专业技巧来确保你可以高效自如地使用它。

本书的读者群

本书是写给那些有过 CSS 编写经验，并希望使自己的网站和技术更上一层楼的读者。我假设你已经知晓了 HTML 和 CSS 的语法和相关术语，但你不必是一名 CSS 专家，而且不需有过使用 CSS3 新特性的经历。无论你是刚接触 CSS，还是已经从事了多年的网站开发工作，这本书都会教给你一些强大的新技巧，把你全面武装起来。

实例文件

每一章都至少包含一个实例演练，使你有机会在真实的页面里一步步地实践技术。你可以从

本书的官方网站 (www.stunningcss3.com) 下载这些文件来配合每一次的实例演练。我为每个实例准备了至少两种文件——初始版和完成版，还为一些篇幅较长的实例提供了中间步骤的版本，你可以定期进行对比来确保在自己的文件上执行了正确的修改。

你可以用任何顺手的编辑器来处理这些实例演练的文件，编写 CSS 无需特殊的工具。我个人使用 Adobe Dreamweaver 的代码视图(Code View)来手工编写 CSS。如果你也使用 Dreamweaver 或类似的编辑器，我推荐你选择手工编写。

尽管已经花费了大量的精力检查过书中的代码，但难免还会存在一些错误。请通过本书网站上的电子邮件地址将错误报告给我，如有必要我会在网站上进行提示并更新实例演练的文件。

实用链接

每一章包含了若干个实用链接，它们是可能对你有用的资源、文章、教程、工具和实例。而且直接点击链接无疑比你费劲对照书本去输入 URL 要来得方便，于是我也将每章的实用链接汇总到了 www.stunningcss3.com 上。

CSS3 是一个多变的话题，所以有时当新近的资源涌现出来时我会更新这些链接列表。你可以在书中这些链接处发现提示，以便从网站上找到最新的信息。

浏览器

书中的实例都已在主流浏览器的最新版本上进行过测试。本书编撰之时，这些浏览器的版本分别是 Chrome 6、Firefox 3.6、IE 8、Opera 10.6 和 Safari 5。这些实例也在 IE 9 和 Firefox 4 的 beta 版[①]上进行了测试，但它们在最终发行版上的效果可能会与书中的描述存在不同。

这些实例在一些至今仍被广泛使用的旧浏览器（比如 IE 7 和 IE 6）上也做了测试。很多情况下，最新浏览器上的 CSS3 效果通常也能体现在相同浏览器的较低版本；即使无法完全体现，页面也依旧可以工作，不仅能用，而且看起来也不错。对于每一项特性，我们将一直尽可能地去寻求变通或备用方案，使之兼容那些不被原生支持的浏览器。

关于具体某个特性被哪些浏览器支持，我在每一章里将所介绍的每个属性或选择器的兼容度整理成表。针对每个浏览器给定“是”、“部分”或“否”。“是”表示该浏览器支持所有语法和行为，它可能会存在一些非常小的缺陷，或与规范所述不太一致，但总体而言是符合规范的。“部分”则意味着浏览器只支持部分的语法和行为，也可能存在重大缺陷或与规范相左之处。

一些 CSS3 属性仅当含有渲染引擎的前缀时才能生效（第 1 章你就将学习这些包含前缀的属性）。我在浏览器兼容表里已经注明了哪些属性需要在哪些浏览器下添加前缀。

为了应对某个浏览器较新的版本支持但老版本不支持的情形，我在兼容表里标注了该浏览器最早开始支持某个属性或选择器的版本号。如果浏览器在最近几个版本中已经支持了该属性或选择器，而且使用不支持该特性的版本的那些用户数量又不大，那么我则不会在兼容表中标注版本

① 本书翻译期间，上述两个浏览器均已发布正式版。

号，你可以认为该浏览器的所有版本都兼容它。

说明 反过来说，当某个属性或选择器最早被支持的时间尤其值得强调时，我也会偶尔标注浏览器的版本号。比如，`@font-face` 早在 IE 4 时代就被支持了。

本书约定

本书使用了一些约定，有必要在这里先说明一下。

- W3C 表示万维网联盟（World Wide Web Consortium），它是一个制定 Web 官方标准和规范（比如 CSS3）的组织。
- IE 是 Windows Internet Explorer 的缩写。“IE 8 及之前的版本”代表 IE 8、IE 7 和 IE 6。
- Webkit 内核浏览器意指 Safari（包含桌面和移动版）、Chrome 和其他使用近期版本的 Webkit 页面渲染引擎的浏览器。
- 偶尔你也会碰到“所有浏览器”这个说法。它代表目前所有被广泛使用的浏览器，而并非字面意义所涵盖的那些可能仅占零星市场份额的不知名的浏览器。

本书的所有示例都以 HTML5 标记编写。但这仅仅表示我使用了短小精悍的 HTML5 文档声明`<!DOCTYPE html>`，还有更简洁的 `meta` 字符编码、`style` 和 `script` 标签。我并没有使用任何 HTML5 引入的新标签，如 `section` 和 `article`，所以页面可以在 IE 8 及之前的版本下正常运行，但你可以在自己的页面里将其更换为喜欢的标记。所有的示例也同样兼容 HTML 4.01 和 XHTML 1。

所有 CSS 实例代码必须置于一个外部样式文件之中，或是 HTML/XHTML 文档的 head 标签中。为了方便修改，这些实例文件本身将 CSS 书写在了页面的 `head` 标签中，但对于最终的成品最好还是将其转移到一个外部文件里。

一些实例代码中含有一些被加粗的字符和行，如果它是你之前看到过的代码片段，则说明这块内容有增改；如果是一段新的代码，则表示它需要你特别注意。某些情况下，你会在某行代码之前看到一个¬符号，它代表这行代码由于版面限制被强制换行了，但这并不意味着你的代码也必须在那里换行。

书中所介绍的每个属性或选择器都有一个题为“来龙去脉”的附注栏，它将简述相关的语法、行为和用例。当然，其中不可能包含所有的语法细节，但一些最根本的信息还是值得参考的。此外，附注栏中还会包含定义相应属性及选择器的 CSS3 模块的站点链接，便于你在想了解完整的规范时参考。

致　　谢

我要感谢所有帮助我最终顺利完成此书编撰工作的人。

尤其要感谢我的文字编辑 Wendy Katz，她确保了我的文字通顺、准确且优雅。与她的再次合作非常愉快。感谢她给予我的指导、建议以及修订意见，特别是不厌其烦地纠正我在使用“仅仅”一词时的语序错误。终有一天我将参透其中的奥秘。

我的作品很大程度上还应当归功于技术编辑 Chris Mills。感谢他一丝不苟地检查代码，提供实用的建议，回答技术问题并指出文中尚需纠正或详述之处。

感谢来自 Peachpit 出版社的组稿编辑 Wendy Sharp，这本书的出版离不开他的辛勤工作。同样还要感谢 Peachpit/New Riders 出版社里所有参与了本书出版工作的其他成员。我不知道他们的名字，但我知道他们付出了很多的体力和脑力，非常感激。

感谢我的文稿代理人 David Fugate 的指导及支持。

许多人并未直接参与本书的编撰工作，但我从他们那里得到了很多素材。感谢 Zoltan Hawryluk、Paul Irish 和 Richard Fink 答复我的技术疑问并提供适用于 CSS3 的优质工具。感谢我的中学老师 Butkus 先生教会我 HTML 和 Photoshop 4，从而带领我走上这条路。感谢我的前任老板 Lamar Heyward 向我的脑中植入了“用 CSS 来实现所有样式和布局”这一疯狂的想法。还有 John Allsopp、Dan Cederholm、Andy Clarke、Chris Coyier、Roger Johansson、Eric Meyer、Trent Walton、Estelle Weyl 等人，与他们共事我学到了很多。他们一直很聪慧，并不吝于与他人分享他们的智慧。

最后，我还要感谢我家庭的每个成员。Cary，感谢你的爱一直支持并伴随着我走完这第二次[①]的著书征途。你用不可思议的耐心长时间陪伴着我，听我抱怨浏览器的缺陷和那些晦涩的 W3C 规范，你还在我长时间写作时将 Asha 照顾得很好。母亲、父亲和 Faith，你们同样在这期间照料过 Asha，非常感激你们的帮助。Asha，也要谢谢你让妈妈得以安心工作，并带着你的“抱抱”定期造访，让我休憩片刻。我爱你们。

---

① 第一次是指 *Flexible Web Design* 的著书过程，中文版《灵活 Web 设计》由机械工业出版社出版。

目　　录

第 1 章 CSS3 的来龙去脉

在使用 CSS3 之前，你应该对它的来龙去脉有个基本了解。在这一章中，你将知晓 CSS3 与 CSS2.1 的区别，以及当前浏览器对 CSS3 的支持情况。对于那些尚未完全支持 CSS3 的浏览器，我们会介绍一些临时兼容的方案和模拟实现的方法。你也会了解到即刻在项目中使用 CSS3 所带来的各种实际好处，包括可以用来说服多疑的客户或老板的一系列理由（这可是实际的好处，别认为它们仅仅只是些理论观点）。最后，我们还会介绍如何使用 CSS3 才能符合渐进增强的设计原则，并讲解一些最佳实践，让应用了 CSS3 的项目尽可能地健壮且与时俱进。

1.1 什么是 CSS3

CSS3 是 CSS2.1 的扩展，它在 CSS2.1 的基础上增加了很多强大的新功能，但是它已不再像 CSS2.1 那样是单一的规范。CSS3 被划分成几个模块，每个模块都是 CSS 的某个子集的独立规范，比如选择器、文本或者背景。每个模块都有各自独立的创作者和时间表。这样做的好处是整个 CSS3 规范的发布不再需要停下来等待某个难产的小条目——这个模块或许尚需等待，但其他模块的流程却能够继续向前推进。

在 www.w3.org/Style/CSS/current-work 这个网页里，你可以看到一份包含了所有模块的列表，以及它们目前的进度离最终发布还有多远。稍后我们将在本章讨论这些模块的当前进度，现在先来看看 CSS3 中激动人心的新特性吧！

1.1.1 新特性概览

理所当然，大部分的 CSS3 规范都重复了 CSS2.1 的内容，但也在它的基础上进行了很多增补和修订。下面列出来的变更并不全面——全部列出也不太现实，因此我们只罗列了从 CSS2.1 到 CSS3 中那些支持度较好、更流行且更实用的变更。

- **不依赖图片的视觉效果**。CSS3 包含了大量新特性，可以用来创建一些以前只能通过图片（或脚本）才实现的视觉效果，比如圆角、阴影、半透明背景、渐变以及图片边框等。在这些新特性之中，多数是属于“背景和边框”（Backgrounds and Borders）模块的，其余的则属于“色彩和图像”（Colors and Image Values）模块。我们将在第 2 章中介绍这些特效，

并在之后的章节里多次应用它们。

- **盒容器变形**。CSS3 中还有一类视觉效果，让我们可以在 2D 或者 3D 空间里操作盒容器的位置和形状，比如旋转、缩放或者移动。这些特效称为变形，在“2D 变形”（2D Transforms）和“3D 变形”（3D Transforms）模块中都有涉及。你将在第 2 章中了解变形特效。
- **独一无二的字体**。“字体”（Font）模块引入了`@font-face`规则，让你能够引入一个存放于服务器的字体文件，并使用该字体来显示页面中的文本，这就突破了以往只能使用用户机器上的字体的限制，也使得页面能呈现出更漂亮的页面。你将在第 3 章详细了解`@font-face`。
- **强大的选择器**。CSS3 新增了十多个选择器，大部分是伪类和属性选择器。你可用它们选取 HTML 结构中的特定片段而无需增加特定的 ID 或类，从而精简代码并使之更加不易出错。这些选择器都描述在“选择器”（Selectors）模块里，在第 4 章和第 5 章我会详细地介绍它们。
- **过渡与动画**。CSS3 的过渡（transition）在其同名的模块里描述。它是一种简单的动画特效，可以平缓地呈现一个元素的样式变化。例如，当用户将鼠标悬停于按钮之上时渐进且平滑地改变其颜色。更复杂的 CSS3“动画”（animation）特性也在其同名的模块里有相应描述，它能够实现更复杂的样式变化和元素位移，而不需要用到 Flash 或 JavaScript。我们会在第 5 章讨论过渡与动画。
- **媒体信息查询**。“媒体信息查询”（Media Queries）模块介绍了如何根据用户的显示终端或设备特征来提供样式，这些特征包括屏幕的可视区域宽度、分辨率以及可显示的色彩数等。媒体信息查询是一款非常棒的专门针对移动设备来实现优化的工具。你将在第 6 章了解到它的相关信息。
- **多列布局**。CSS3 引入了几个新模块来帮助我们更方便地创建多列布局。“多列布局”（Multi-column Layout）模块描述了如何像报纸布局那样把一个简单的区块拆分成多列，我们将在第 6 章对它进行讲解。而“弹性盒容器布局”（Flexible Box Layout）模块则能够让区块在水平或垂直方向上保持对齐，相对于浮动布局或绝对定位布局来说它显得更为灵活。此外还有“模板布局”（Template Layout）和“网格定位”（Grid Positioning）的实验性布局模块，我们将在第 7 章讨论后面 3 种布局系统。

1.1.2　CSS3 的当前进展

我已经能想象到你要问：这些相当酷的新特性何时才能最终发布呢？正如我在之前的章节里所提到的——每个模块都有它自己的时间表，你可以到 www.w3.org/Style/CSS/current-work 网站了解其进展（图 1-1）。这张表格上所列出的状态，通常被称为*成熟级别*，有时候 W3C 也称其为*稳定级别*，这些模块的当前版本及下个版本的状态都已链接到对应的文档。

High Priority	Current	Upcoming
CSS Level 2 Revision 1	Candidate Recommendation	Proposed Recommendation
Selectors	Proposed Recommendation	Recommendation
CSS Mobile Profile 2.0	Candidate Recommendation	Proposed Recommendation
CSS Marquee	Candidate Recommendation	Proposed Recommendation
Medium Priority	**Current**	**Upcoming**
CSS Snapshot 2007	Last Call	Candidate Recommendation
CSS Namespaces	Candidate Recommendation	Proposed Recommendation
CSS Paged Media	Last Call	Last Call
CSS Print Profile	Last Call	Candidate Recommendation
CSS Values and Units	Working Draft	Working Draft
CSS Cascading and Inheritance	Working Draft	Working Draft
CSS Text	Working Draft	Working Draft
CSS Text Layout		Working Draft
CSS Line Grid		Working Draft
CSS Ruby	Candidate Recommendation	Working Draft
CSS Generated Content for Paged Media	Working Draft	Working Draft
CSS Backgrounds and Borders Level 3	Last Call	Candidate Recommendation
CSS Fonts	Working Draft	Last Call
CSS Basic Box Model	Working Draft	Working Draft
CSS Multi-column Layout	Candidate Recommendation	Proposed Recommendation
CSS Template Layout	Working Draft	Working Draft
Media Queries	Candidate Recommendation	Proposed Recommendation
CSS Speech	Working Draft	Working Draft
CSS Color	Last Call	Proposed Recommendation
CSS Basic User Interface	Candidate Recommendation	Test Suite
CSS Scoping		Working Draft
CSS Grid Positioning	Working Draft	Working Draft
CSS Flexible Box Layout	Working Draft	Working Draft
CSS Image Values	Working Draft	Working Draft
CSS 2D Transformations	Working Draft	Working Draft
CSS 3D Transformations	Working Draft	Working Draft
CSS Transitions	Working Draft	Working Draft
CSS Animations	Working Draft	Working Draft
Low Priority	**Current**	**Upcoming**
CSSOM View	Working Draft	Working Draft
CSS Extended Box Model		Working Draft
CSS Object Model		Working Draft
CSS Syntax	Working Draft	Working Draft
CSS Lists	Working Draft	Working Draft
CSS Tables		Working Draft
CSS Reader Media Type	Working Draft	-
CSS Positioning		Working Draft
CSS Generated and Replaced Content	Working Draft	Working Draft
CSS Line Layout	Working Draft	Working Draft
CSS Hyperlink Presentation	Working Draft	Working Draft
CSS style Attribute Syntax	Last Call	Candidate Recommendation
CSS Math		Working Draft
CSS Presentation Levels	Working Draft	Working Draft
CSS Aural Style Sheets		Working Draft
CSS TV Profile 1.0	Candidate Recommendation	Proposed Recommendation
Behavioral Extensions to CSS	Working Draft	Working Draft
CSS Introduction	Working Draft	Working Draft
Other	**Current**	**Upcoming**
SVG	Recommendation	Recommendation

图 1-1　CSS3 当前所有的模块及其状态

W3C 所使用的级别，从低至高依次如下所示。

(1) **工作草案**（Working Draft）。这是最早的可公开版本，发布的目的是为了能从社区征集更多的意见，一个模块或规范可能会历经若干个工作草案。

(2) **最终公示**（Last Call）。它相当于带有最后评述期限的工作草案。这表明工作组认为该模块已经实现了既定功能——尽管通常在此之后还会发生一些重要的变更，而且它极有可能正准备开始进入下一个阶段了。

(3) **候选推荐标准**（Candidate Recommendation）。工作组认定该模块已经稳定并符合使用需求

了，浏览器应该可以着手实现本模块了；而为了了解如何部署该模块，Web 开发者也可以在日常开发中使用这些特性了。浏览器在实现这些特性时可以带上各自的私有前缀。在这之后规范仍有可能发生变更，但不会很频繁或很重要。

(4) 提名推荐标准（Proposed Recommendation）。这将会是非常成熟且经过认真审核过的文档，它将被提交到 W3C 顾问委员会做最后的批注。在这之后鲜有变更。

(5) 推荐标准（Recommendation）。这是最终的完整版本，即我们通常所指的“标准”。

Web 开发者显然很希望在 W3C 规范完成之前就可以使用这些新特性，而事实上，他们也期待我们这样去做。如果你认真查看了 W3C 网站上的列表（如图 1-1 所示），就会注意到在本书撰写之时仅仅只有“SVG”模块（它在列表靠近底部的位置）是处于“推荐标准”状态的。即便是我们已经使用了很多年的 CSS2.1，目前还依然处于“候选推荐标准”的状态，甚至尚不属于“提名推荐标准”。因此，即使 CSS3 还不是一个最终的标准，我们还是可以大量地使用它。

1.1.3　现在就开始使用CSS3

有几个 CSS3 规范模块目前已处于“候选推荐”状态了，这说明它们已经完全可以正常使用了。即便是尚处于“工作草案状态”的某些特性，我们也可以去用一用。虽然还需要等待某些尚在变化中或支持度不足的属性和特性变得更稳定些，但在合适的情形下，我们不妨去使用 CSS3 里那些支持度较好且相对稳定的部分。

只有将新的 CSS 技术运用到实际工作中，我们才能够发现应用这些技术所面临的真正挑战，以便帮助 W3C 来应对它们。在真实环境下使用新的 CSS 技术，能够帮助 Web 开发社区发现其缺点、差异以及规范中的漏洞，而且能够引入一些关于改进、扩充和阐明规范的新思想。当我们还有机会改变 CSS3 规范的时候，可以通过测试来帮助它变得更好，所以请不要坐等标准的最终确立而错失良机。

实际应用这些从某种意义上说尚处在最前沿的技术也能够告诉浏览器制造商：对于 Web 开发者来说，哪些 CSS3 特性最流行且最实用。这也能促使浏览器制造商更好地支持这类新特性。

因此，尽早地使用新的 CSS 特性能够有效促使它们成为真正的标准。如果它们从未被使用过，那么永远也不可能被最终确立。

我并不是说 W3C 网站的那个列表上的所有内容都能马上投入使用。并不是所有的新特性和技术都能够立即开始使用，或是被应用到所有的项目里。你应该只使用那些相对稳定的 CSS3 特性，并确保不会对那些尚不支持这些特性的浏览器造成影响。明智地使用它们！不要仅仅是因为能使用这些特性就无限制地滥用，要看看这些特性是否符合网站的目标，以及是否对用户有益，然后才在适当的地方使用它们。

有些 CSS3 特性还没有达到“候选推荐标准”的级别，但是它们的语法已经十分稳定，在很长一段时间之内都没有变化，而且未来也不太可能发生变更。但遗憾的是，单从 W3C 的网站来看，你无法知晓哪些特性符合这个条件，因此必须依赖其他文章或者书籍来确定某个特性的历史和稳定程度。在本书中，我们将接触到几乎所有稳定且适合使用的 CSS3 特性；而极少数情况下当深入使用更具试验性质的特性时，书中会予以提示。

1.2 浏览器的支持状况

另一个值得考虑的问题是，通常我们需要知道各大主流浏览器或某类特定用户所使用的浏览器对某个 CSS3 特性的支持程度如何。有时候花了时间去增加一段还在实验阶段且支持度不佳的 CSS——比如说为某个特定的浏览器增加了一个小“彩蛋”，却只能被一小部分用户欣赏到，那么这样的工作通常并没有什么实际意义。

然而为了知道你的大部分用户能够体验到哪些 CSS3 的新特性，就必须了解当前浏览器对这些特性的支持程度。

1.2.1 浏览器市场份额

浏览器的使用率始终在变化而且难以精确掌握，图 1-2 显示了 2010 年 10 月各大浏览器的市场份额。这些图表根据 Net Applications 的可信度高且样本广泛的统计结果绘制而成（http://marketshare.hitslink.com/browser-market-share.aspx?qprid=0）。如果想了解其他来源的统计结果，可以访问维基百科的“Usage share of web browsers”页面[①]（http://en.wikipedia.org/wiki/Usage_share_of_web_browsers）。

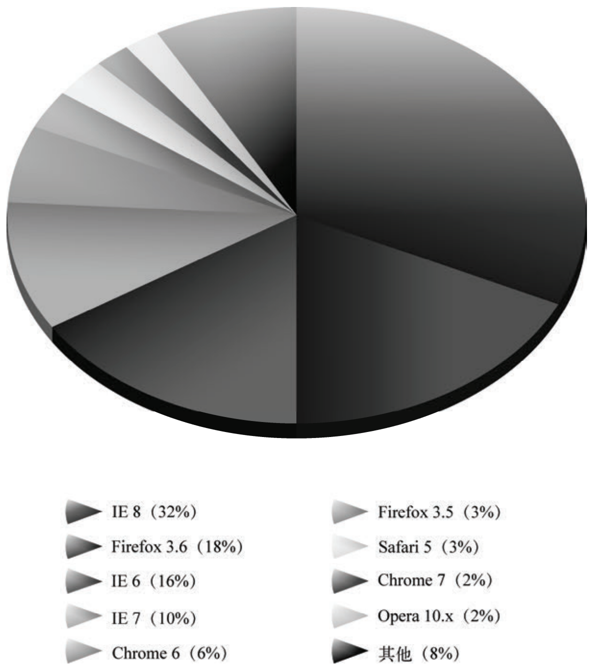


图 1-2 2010 年 10 月的浏览器使用率（另见彩插图 1-2）

① 该词条在维基中有中文版“网页浏览器的使用分布”，可从英文页面中进行切换。

请注意 IE 6 的市场份额已经变得相当小了[①]（此结论相对于它在 2006 年年初达到顶峰时的份额来看，当时为 85%，而且 IE 5.x 版本也占据了 5%），但它也不能被完全忽视，目前 Firefox 3.6 的用户数仅略多于 IE 6。如果把其他版本的 Firefox、Chrome、Safari 和 Opera 加在一起计算的话，IE 6 的市场份额确实已经远远少于这些符合标准的现代浏览器。通常，当某人在博客上展示一个很酷的 CSS 技巧的时候，我们首先会想到的问题是："它在 IE 6 上看起来咋样？"然而现在，根据当前的浏览器使用率统计结果，是不是询问"它在 Firefox 上看起来咋样"显得更有意义呢？

这并不意味着你可以忽略针对 IE 6 的测试，或是干脆禁止这类用户访问你的网站。网站的内容始终应该在任何浏览器上都是可用的，制作一个令人满意的 Web 页面并让它在 IE 6 上易于使用并不是一件很难的事情，但是为了一个逐渐消失的用户群体花费大量时间和精力确实是越来越无用了，除非你的目标用户群很多仍在使用该浏览器。正如我先前所言，CSS3 对你的网站到底意义几何，是由你的项目和用户所决定的，而不是这个大致的浏览器使用率统计结果。

但是，除非你的网站统计结果与这个结果的出入很大，否则就不该继续认为非 IE 用户仅仅是个不需要被特别关注的边缘化群体。花费在非 IE 浏览器上的时间与花费在 IE 6/7 上的时间同样重要。读过本书，你会发现使用 CSS3 可以很容易地让网站在非 IE 浏览器下看起来更棒——少数情况下甚至对 IE 也适用。

1.2.2　主流浏览器支持状况

幸运的是，大部分我们想要使用的 CSS3 特性都已经有了很好的浏览器支持度。在讲解每一个 CSS3 属性、特性或选择器时，我都会提及相应的浏览器支持度，但一份总的浏览器支持度概览也会相当实用。下面我就为各浏览器对 CSS3 新特性的总体支持程度做一个评级。

说明　浏览器支持度信息见附录。

- **Safari 5、Safari for iOS4 和 Chrome 5：B+**。虽然 Safari 和 Chrome 是不同的浏览器，渲染 CSS3 属性所用的方法也不尽相同，但是它们都使用相同的 Webkit 渲染引擎，并且对 CSS3 的支持程度也几乎一致。它们支持大部分的 CSS3 新特性，甚至包括其他浏览器所不支持的"动画"特性。Safari 5 是唯一支持"3D 变形"的浏览器。它们主要的问题是其"渐变"特性的语法是非标准的，而"多列布局"和"弹性盒容器布局"模块的实现也是不完整且存在缺陷的。它们同样不支持"模板布局"和"网格定位"的模块，但目前也没有其他浏览器能做到这一点。
- **Safari 4、Safari for iOS3：B**。iOS 是 Apple 的移动操作系统，在 iPad、旧版 iPhone 和 iPod Touch 以及尚未更新系统的新版 iPhone 和 iPod Touch 上运行的是它的第 3 版。iOS3 里的

---

① 文中数据代表全球范围。据各大互联网企业的统计，IE 6 至今仍占据国内浏览器市场的半壁江山。

Safari 与桌面版 Safari 4 的功能基本一致。Safari 4 和 Safari 5 对 CSS3 的支持度基本处于同一个级别，只是不支持“3D 变形”，另外还有一些属性存在细微的语法问题。

- Firefox 4：B+。除了不支持“动画”和“3D 变形”之外，Firefox 4 的支持度与 Safari 5 几近相同，但它用更符合标准的“渐变”语法、更好的“多列布局”模块的实现以及独家支持的“`calc` 函数”弥补了“不足”。然而它对“弹性盒容器布局”模块的实现同样不完整且存在缺陷。
- Firefox 3.6：B。Firefox 3.6 的支持度和 Firefox 4 大体一致，但不支持“过渡”和“`calc` 函数”。
- Opera 10.6：B。Opera 10.6 能够支持 Firefox 3.6 所能支持的绝大部分特性，除了“渐变”和“弹性盒容器布局”。但它支持“过渡”，这是 Firefox 3.6 所不支持的。
- Opera 10.5、Opera Mobile 10.1：B-。Opera Mobile 10 对应于桌面版的 10.5，它们的支持度与 Opera 10.6 总体一致，但在一些属性上缺陷略多。
- IE9：C+。在本书完稿之时 IE 9 尚处于 beta 阶段[①]，它大概只支持其他主流浏览器所支持的特性的一半。其优点是对所支持的特性实现度非常好，且不需要使用浏览器前缀（稍后在本章你将会了解到浏览器前缀的相关内容）。
- IE 8、IE 7 和 IE 6：D-。不用说，由于这些浏览器都发布于 CSS3 完善发展之前，因此它们几乎完全不支持 CSS3。然而它们全都支持`@font-face` 和 `word-wrap` 特性。其中 IE 7 和 IE 8 还支持 CSS3 的属性选择器。除此之外 IE 8 还支持 `box-sizing`。

其他浏览器支持信息资源

在本书中我会提供很多详细的浏览器支持度信息，但 CSS3 的浏览器支持度一直在持续变化之中。本书无意成为关于 CSS3 属性、值、函数、规则和选择器的百科全书，所以你可能需要到其他地方查阅 CSS3 某个特性的浏览器支持度。

Mozilla、Opera 和 Safari 为它们所支持的特性维护着非常实用的列表：

- https://developer.mozilla.org/en/Mozilla_CSS_support_chart
- http://www.opera.com/docs/specs
- http://developer.apple.com/safari/library/documentation/AppleApplications/Reference/SafariCSSRef

其他关于浏览器支持度的网站如下所示。

- 维基百科的“Comparison of layout engines（Cascading Style Sheets）[②]”（http://en.wikipedia.org/wiki/Comparison_of_layout_engines_（CSS））页面提供了如你所想的那般全面而翔实的内容。

---

① 本书的译稿完成之时，IE 9 已经发布了正式版本，对 CSS3 特性有了非常好的支持。

② 该词条在维基中有中文版“浏览器引擎 CSS 支持比较”，可从英文页面中进行切换。

- FindMeByIP（www.findmebyip.com/litmus）整理了许多浏览器对一些主要的 CSS3 属性和选择器的支持情况，此外也包括 HTML5 的特性。
- “When can I use...”（http://caniuse.com/#cats=CSS3）列举了五大浏览器在当前、过去以及未来版本中对若干个广受欢迎的 CSS3 属性和特性的支持度信息。
- Standardista（www.standardista.com/css3）目前包含了对 `borders`、`backgrounds`、`columns`、`@font-face` 和选择器等特性的详细支持图表。更多模块将被定期添加进去。
- QuirksMode 在“CSS contents and browser compatibility”（CSS 内容和浏览器的兼容性）页面（www.quirksmode.org/css/contents.html）里罗列了各种 CSS3 和 CSS2.1 选择器和属性的支持度信息，其中每一条都有独立页面来详细地介绍它的工作原理，而实际上浏览器又是如何去处理它的。
- CSS Infos 维护着一张 `-moz-` 和 `-webkit-` 的私有属性列表，它展示了每个属性诞生时的浏览器版本。请参阅 http://css-infos.net/properties/firefox.php 和 http://css-infos.net/properties/webkit.php。
- Campaign Monitor 则提供了一份电子邮件客户端对 CSS 的支持度指南，其中也包含 CSS3 的属性，这样一来你就能知道在 HTML 格式的电子邮件时事通讯里什么可用什么不可用了。请参阅：www.campaignmonitor.com/css。

这些级别各异的浏览器支持度意味着你的页面在不同的浏览器中可能看起来不太一样。这没什么，而且坦白讲，无论你是否使用 CSS3，这种情况都将不可避免地存在。因为终端媒体天生就具备用户可控的性质，所以 Web 页面从来都无法保证在任何地方都表现一致。而且在如今这个时代，设备、屏幕、浏览器的形态越来越丰富，人们的惯用设置也不尽相同，因此想要创造一个在任何地方都表现一致的页面就更加不可能了。

只要你关注于向所有人提供实用且易用的页面，那么表现上的差异就没那么重要。而这就是“渐进增强”哲学的思想所在。

1.3　渐进增强

渐进增强（Progressive Enhancement）是这样一种开发方式：在你编写 Web 页面时，首先让它们在基础的浏览器和设备上正常工作且展现得当，然后再辅以更高级但非必要的 CSS 和 JavaScript 等增强功能，来为当前和未来的浏览器提供更好的支持。举例来说，你可以用朴素但语义化的 HTML 构建一个表单，即使没有样式它也能得到良好的展现，禁用 JavaScript 后它也可以正常工作；随后你可以用 CSS 来丰富其外观，再使用 JavaScript 增加客户端层面的验证。我们的目标是，利用现代浏览器的优势特性，在确保网站能适用于每个用户的同时去创造尽可能丰富的体验。*Designing with Progressive Enhancement*[①]（http://www.filamentgroup. com/dwpe/）一书对

① 本书中文版即将由人民邮电出版社出版。——编者注

此有很好的总结：

> 渐进增强……其目标是向尽可能广泛的受众提供尽可能优质的体验。跟用户访问网站所采用方式无关，无论是通过 iPhone、高端的桌面系统、Kindle，还是读屏器，他们所能得到的体验应该尽可能的独特且完备。

1.3.1 优点

虽然“向尽可能广泛的受众提供尽可能优质的体验”这一伟大目标听起来相当不错，但其实不使用渐进增强技术你也可以实现它，比如为旧式浏览器提供一套兼容方案来确保页面与其在新式浏览器下的表现和行为尽量保持一致，但这通常并不是明智的选择。改用渐进增强技术能够让网站的视觉丰富程度随着浏览器的现代化发展而不断提升，用户和你都能从中受益。

优雅降级

你或许对优雅降级（Graceful Degradation）这个术语非常熟悉，并且认为它与渐进增强有着相同的含义。但实际上它们是两种完全相反的处理方式，尽管通常会带来同样的效果。使用优雅降级技术时，你必须首先完整地实现了网站，其中包括所有的功能和效果。然后你再为那些无法支持所有功能的浏览器增加候选方案，使之在旧式浏览器上以某种形式降级体验却不至于完全失效。

而使用渐进增强时，无需为了让一个已成型的网站在旧式浏览器下正常工作而做逆向工程。首先，只需要为所有的设备和浏览器准备好清晰且语义化的 HTML 以及完善的内容，然后再以无侵入（unobtrusive）的方式向页面增加无害于基础浏览器的额外样式和功能。当浏览器升级时，它们会自动地呈现出来并发挥功用。本书示例中有渐进增强技术的诸多实践，每个示例页面首先都能正常地显示并工作，然后再通过 CSS3 使之得到增强。

首先，为了在支持度较低的浏览器也能实现相同的效果而添加 Hack、兼容方案、模拟性脚本或其他技术将会耗费大量的人力和时间。即使最终你（在所能测试的有限数量的用户环境之下）实现了近乎一致的表现，但问题是用户真正收获了什么？与其试图让 IE 看起来堪比年轻它十岁的浏览器，不如努力改善网站的可访问性，或是进行更多的可用性测试，或是去实现其他真正可以帮到用户的改进工作，而不仅仅是让页面看起来“更靓一点”。

除此之外，如我之前所言：想让网站在任何环境下看起来都保持一致是不可能的，不管为此付出多少努力，结局依旧会令你失望。因此，如果网站无论如何都看起来不同，那么为什么不用 CSS3 技术来使它在新式浏览器上看起来更为惊艳呢？某些 CSS3 特性在不支持它的浏览器中简直是“无法模拟实现”的，但若使用渐进增强，就无需为了能让你的网站适合所有人而放弃这些技术。仅仅因为部分人不愿或不能升级浏览器，却让使用新型浏览器的用户无法享受 CSS3 所提供的伟大技术，这是毫无道理可言的。我们应该先让网站能够正常工作于尽可能旧的浏览器上，然后不断为它在新型浏览器上实现更多的增强和改进。这个方法可让每个用户都

得到对他来说最好的体验。随着时间的推移，当越来越多的人开始升级浏览器而浏览器本身的支持度也不断提升时，就会有越来越多的人体验到这些增强和改进，它持续有效地使网站越来越好，却无需你刻意做什么。只需要一次实现，它就让网站的体验与时俱进。

大部分人可能永远都不知道你的网站在不同的浏览器和设备上看起来并不相同，因为普通人并不会像我们这群“偏执”的 Web 设计师一样使用多个浏览器来对比设计的细节差异。即使他们会使用不同浏览器来访问你的网站，只要这些差异不会影响网站的易用性（如果你处理得当的话），他们也不会对此有什么特别的想法。说到底，假如有人上班时使用 IE 8，回家后使用笔记本上的 Chrome，在 iPhone 上使用 Safari，并且在 Wii 上使用 Opera 来访问你的网站，我想他终究会习惯所有这些不同设备上的不同表现。

1.3.2 “让我来打个比方……”

我是个极喜欢“打比方”的人，不仅是在日常生活中，工作中也如此。我觉得用这种方式为客户解释技术概念或是证明我所做的可用性改变的重要性确实挺不错。因此，尽管你已经开始使用渐进增强技术，你还是可以使用以下这些比喻来说服尚存犹疑的客户或老板。

假设你让亲爱的老公（老婆）下碗面。面被端到桌前时，它已具备了所有能让我们得以称其为“面”的食材：筋道的面条、鲜美的面汤、适量的盐，甚至还可能有葱花儿什么的。这面尝起来还不错，符合你的要求。

但你怎么都想不到，隔壁邻居吃的面不仅具备你的所有食材，同时还搭配有榨菜、肉末、炸酱，甚至还煎了一个荷包蛋，这些都不是一碗面所必需的，但它们能让这碗面变得更加美味。

这例子用在 Web 网站上也一样：一个简洁清晰并能兼容 IE 6 的网站就是一碗普通的面，它完全能符合访问者的期望，也可以满足他们对内容的需求。IE 6 用户不会想到 Firefox 用户竟可以看到一些用 CSS3 所实现的更为别致的东西，除非你的网站用旧式浏览器看起来就是乱七八糟的（就像是面里没放盐）——你最好修复这个问题——否则你的用户想不到换个浏览器就会别有洞天。

如果你是平素很少吃面的南方人，也可以想象一下一碗颗粒饱满却平淡无味的白米饭与另一碗光润油亮且色味俱全的扬州炒饭之间的差异。又或者你平时爱看电视，那么一台小型的传统电视和一部高清纯平的液晶电视也能产生相同的对比效果——液晶电视的效果显然会悦目很多。想让陈旧设备下的页面看起来也一样其实是很不明智的，要知道 IE 6 可是 2001 年的产品。

另外一件很不明智的事情是期望录像机能够播放蓝光碟。这从一开始就不可能，因为录像机的发明比蓝光早很多年。蓝光碟使用了很多新技术来提高画质和功能，这远不是录像带所能提供的。只要你愿意，你依旧可以用录像机来看电影，但是蓝光版本的电影看起来更棒，还提供很多额外内容。每个人都看到了相同的电影内容，但新技术的驾驭者立即获得了额外的好处，而无需等到录像机退出历史舞台。

同样道理，当有人在展示一项 CSS3 技术的时候去询问其在 IE 6 上的表现也是毫无意义的，

它比 CSS3 的发明都早出现了好些年。只要用户所获得的内容都是相同的，而且 CSS3 技术对网站在 IE 6 下的表现也毫无损害，那么这项仅有部分人可以享受到的设计理念就值得肯定。

1.4 CSS3 的好处

我希望你已经清楚地了解了为什么渐进增强这项通用的开发技术非常好，但我们还没有开始讨论 CSS3 的独特好处。CSS3 的优点远远不止于能让页面看起来酷炫异常——尽管这也是不可忽视的。当你读过本书之后就会知道，CSS3 能帮助你创建很多十分炫目的效果，使网站设计锦上添花。

但其实大部分的 CSS3 视觉效果是无需依赖 CSS3 来实现的，你还可以找到很多替代的方法，比如使用图片、JavaScript 或 Flash。因此除了“看起来更酷”之外，还需要有其他的理由来支持我们使用 CSS3。

这种原因的确存在——总之，在大多数情况下，使用 CSS3 不仅可以易于开发和修改页面，还能减少这个页面的加载时间。与此同时，你还能增加网站的可访问性和可用性，使网站能够适配更多的设备，甚至还可以优化网站的搜索结果排名。下面更详细地介绍这些好处。

更多关于渐进增强的信息

关于渐进增强实在是有太多可以说的了，事实上，甚至有一整本专门用来讲述它的书，名叫 *Designing with Progressive Enhancement*（www.filamentgroup.com/dwpe）。尽管我认为我已经描述得足够详细了，但你可能还需要更深入地了解什么是渐进增强，以及为何它关乎能否说服你的团队成员、老板或客户。因此我希望下列这些链接能对你有所帮助。

- *The Case for Designing with Progressive Enhancement*（渐进增强设计实例），作者是 Todd Parker、Maggie Costello Wachs、Scott Jehl 和 Patty Toland（www.peachpit.com/articles/article.aspx?p=1586457）。
- *Progressive Enhancement: What It Is, And How To Use It?*（渐进增强的概念和用法），作者是 Sam Dwyer（www.smashingmagazine.com/2009/04/22/progressive-enhancement-what-it-is-and-how-to-use-it）。
- *Progressive Enhancement: Paving the Way for Future Web Design*（渐进增强：未来的 Web 设计之路），作者是 Steven Champeon（www.hesketh.com/publications/articles/progressive-enhancement- paving-the-way-for）。
- *Graceful Degradation Versus Progressive Enhancement*（优雅降级和渐进增强），作者是 Christian Heilman（dev.opera.com/articles/view/graceful-degradation-progressive-enhance）。

1.4.1 减少开发和维护的耗时

很多 CSS3 技术通过提供相同的视觉效果而成为了图片的“替代品”。举例来说，假设要为

某个区块实现一个投影效果，你不必再通过创建一个或多个图片作为背景来实现该需求；现如今你只需使用 CSS3 的 `box-shadow` 属性就可以产生阴影效果，这将把你从绘图、切图和优化图片的工作中解放出来。

如果你需要后续地调整投影效果或单纯地想要测试不同参数配置下的效果变化，那么使用 CSS 也比使用图片更为方便。比如你的用户希望使投影更模糊些，或是距离盒容器更远一点，或是用红色替换掉灰色等，当使用 CSS3 的时候你能够在几秒钟之内就实现这些不同的配置，而不需要到 Photoshop 里去执行修改并重新输出图片。

一些 CSS3 技术同样能让你远离脚本和 Flash，这会有效地提升效率，因为你无需花费时间去寻找合适的脚本并修改它以适配你的网站。

最后，有些 CSS3 技术还能让你的页面结构更“苗条”，因为你不用再嵌套很多的 `div` 和类，这也可以被解读为可以有效地减少工作时间。比方说，现在你可以把多个背景图片的定义写在同一个元素当中，这样你就不用去嵌套多余的元素并为它们分别指定不同的背景了。你还可以根据元素在文档树中所处的位置使用 CSS3 的选择符来为它定位，这样一来你不用再花时间去创建更多的类并将它们应用到必需的元素上，之后还要确保新增内容也同样应用了这些类。

1.4.2　提升页面性能

更少的标记和图片意味着用户需要下载的内容将会变少，页面加载得也会更快。更少的图片、脚本和 Flash 文件也就意味着更少的 HTTP 请求数，这就是提升页面加载速度的最佳方法之一。事实上，雅虎专业的性能团队将“减少 HTTP 请求数”视为“关于提升游客首次访问速度的最重要的性能优化指导”（http://developer.yahoo.com/performance/rules.html）。

物尽其用

在本书的许多地方我都提到了 CSS3 技术是如何去取代图像、JavaScript、Flash、类和 div 的多层嵌套的，但这并不是说这些东西原本就是不应该存在的。比如，我没有说应该禁止使用所有图片，那无疑是相当荒唐的。只要用得适当，所有这些东西都还是优秀的工具，而当另一项技术更适合用来解决当前需求时，使用 CSS 取而代之反而是不明智的，比如说不能仅仅因为 CSS 比较“酷”，就放弃实际上更为适用的 JavaScript 技术而改用 CSS 来实现下拉菜单。不过要是使用 CSS3 可以更有效率或能够提升用户体验，同时又可以达到相等甚至更好的表现时，我认为这种选择将是明智的。

当浏览器从服务器获取页面内容时，它需要发起 HTTP 请求。每次浏览器获取 Web 页面中一个不同的内容时，诸如样式、图片、脚本之类的文件，它就必须回到服务器去获取这些特定的文件。这个过程是需要花费时间的，相比所有内容的总字节数而言，它对页面加载速度的影响会更大。这就意味着，一个页面中如果有 10 个 10KB 大小的图片，那么加载这个页面所需的时间将会远大于只有 1 个 100KB 大小图片的页面，甚至可能大于只有 1 个 200KB 大小图片的页面。

使用 CSS3，你就可能创造一个无需任何图片的图形化网站，极大地减少 HTTP 的请求数量，并且能提升页面的加载速度。

当然我并不是说添加的任何 CSS3 特性都会让页面更快，它取决于用 CSS3 来替代什么技术，同时也取决于如何使用 CSS3。

例如第 3 章将介绍如何使用`@font-face` 链接一个外部的字体文件，字体文件本身就会导致一次新的 HTTP 请求——用户必须通过它来拉取字体文件，而且这个文件尺寸还可能会很大。因此在某些情况下，使用`@font-face` 会降低页面加载速度。从另外一方面来说，如果用大量的图片来实现特殊文字效果，那么让用户下载一个字体文件反而会更快。它可能也会比使用 JavaScript 或 Flash 实现文本效果更快。这只是诸多例子中的一个，而页面速度到底如何变化则取决于用 CSS3 来取代什么技术，所使用字体的尺寸大小，是否精简了字体文件中的字符数，以及在具体实现`@font-face` 这个特性时的其他影响因素。

某些图形化的 CSS3 技术，比如渐变，能够减少 HTTP 请求，但可能要求浏览器执行更多的工作来渲染特效，这将可能导致浏览器响应缓慢，并减少可用性。因此，请不要滥用复杂的特效，并要完整地测试所实现的那些特效。

但关键的是，很多 CSS3 技术能够在任何情况下都大幅提高页面的性能。单这一条就足以成为使用 CSS3 的理由，因为用户确实在意页面的加载速度。最近 Bing 和 Google 都在做一些相似的实验：故意将服务器的响应时间随机延迟数毫秒，以此来观察是否对用户体验产生影响。通过实验他们发现：等待时间越长，用户在这个页面上操作的可能性就越少，通常他们会减少搜索次数或者点击次数，然后就会离开这个页面。甚至少于半秒的延迟也会造成很大影响。关于页面迟缓所造成的商业影响，可以参见 Stoyan Stefanov 的"The performance pitch"（"性能商业观点"）（www. phpied.com/the-performance-bussiness-pitch）。

1.5 更好的搜索引擎优化

快速的页面不仅能吸引用户，同时也会让 Google 很高兴，难道我们不都希望在 Google 上有好的表现吗？从 2010 年 3 月开始，Google 开始通过将页面速度作为评分的标准之一来鼓励页面提速，因此加载速度较快的页面在搜索结果中的排名比其竞争者更靠前。

即使 Google 不把速度当作评分依据，并且别的搜索引擎也没有这么做，你依然可以获得很好的搜索引擎优化，只要用 CSS3 美化的正常文本替代图片或者 Flash 即可。虽然搜索引擎也能读取图片的 `alt` 属性或者某些 Flash 文件，但正常的文本通常能让你的网站获得更高的权重。

1.5.1 提高可用性和可达性

使用真实文本而非图片有一个更大好处，即真实文本对任何人来说都更可用，尤其是对残障人士来说。用户可以把真实的文本放大和缩小并能改变颜色，从而使页面更加容易阅读，也更便

于复制粘贴，通过浏览器的查找功能来搜索页面会更容易，通过搜索引擎或者其他工具抓取页面更轻松，也很容易将页面翻译成不同的语言。

CSS3 并不是提高可读性的万能工具，如果滥用样式或者设计，甚至有可能会使页面难以辨认。但如果在真实的文本上正确地使用 CSS3 特性，如`@font-face`、`text-shadow`、变形和其他 CSS3 效果，而不是使用图片，将会极大提高网站的可用性。

另外一个通过使用 CSS3 来提高可用性的例子就是使用媒体查询。之前我已经提过媒体查询如何基于用户的显示特性来定义不同的样式，做到根据用户不同的设备和设置来调整样式。这种技术能够保证最大限度地利用空间，也能让页面在不同的用户浏览场景都具有最佳的可读性。第 6 章会更详细地介绍媒体查询。

1.5.2　与时俱进

学习和使用 CSS3 还有一个额外的好处：它能让你始终处于 Web 设计的前沿。CSS3 不会消失，它是未来我们构建网站的方式。学习 CSS3 越来越重要，这也能够增加你的职业技能和竞争力。现在，它还能帮助你缩短与顶级设计师或开发者的差距。马上开始使用它（至少在个人项目上），然后不断磨砺你的技巧并且在职业道路上不断前进。

1.6　案例学习：公路安全研究中心

为了对这些使用 CSS3 的好处有更好的认识，我们来看看一个真实的网站是如何通过使用 CSS3 来替代旧的 Web 设计从而获得明显提升的。我选择在自己的而不是陌生人的网站上找茬。

1.6.1　使用CSS3 之前

2006 年，我为 UNG 公路安全研究中心（www.hsrc.unc.edu）设计和开发了使用 CSS 以及 HTML 的网页。图 1-3 展示了 HSRC 的主页。自我最开始创建它以来，它基本上没有什么变化，甚至比某些内页还要简单，所以基本上和大部分的网页比较类似，但即便如此，在这个简单的页面上还是有大量的图片存在。你可以看到它使用了很多的圆角、渐变和阴影。

我想观察一下在使用了这些图片之后，这个网站的实际性能如何，所以我在 Firefox 3.6、IE 8 和 IE 6 里面打开它进行了测试，表 1-1 显示了 HTTP 请求的数目和在每个浏览器中的平均加载时间。

表1-1　原始页面的性能

	Firefox 3.6	IE 8	IE 6
HTTP请求	36	37	47
页面加载时间（单位：秒）	1.5	1.3	3



图 1-3　公路安全中心首页（另见彩插图 1-3）

我想这些加载时间还不错，但是它们肯定还能优化。特别是在 IE 6 下，等待的时间有点长。如果我能够减少 HTTP 请求的数量，那就能显著减少页面加载的时间。

大量的 HTTP 请求来源于选项卡式导航条。每一个导航条都是一个单独的图片，并且包含三种状态：非选中态、悬停态和当前选中态（图 1-4）。我最初制作这个页面时，使用了名为“CSS Sprites”的背景图片技术将多张背景图片合并成一张，并通过使用 `background-position` 属性来移动可见的区域。但我并没有过度使用这一技术。



非选中态

悬停态

当前选中态

图 1-4　每个导航条有三种状态

我不太希望拿这个版本来和新的 CSS3 技术进行比较，所以我把所有的选项卡图片都合并成一张大的图片，并且修改了使用这些图片的 CSS，然后对这个修订版本进行测试后，表 1-2 显示了如下测试结果。

表1-2　改进后的页面性能

	Firefox 3.6	IE 8	IE 6
HTTP请求	29	30	33
页面加载时间（单位：秒）	1.3	1.15	2
加载时间减少量	13%	11%	33%

将 HTTP 请求合并无疑能够很好地减少页面加载时间，特别是在 IE 6 下。但是请记住，将所有选项卡图片合并成一张是很困难的工作，而且相对独立的图片来说，后期维护工作也比较复杂，CSS 也会变得很复杂。这就是你使用 Sprites 技术的代价。但是拿这个页面与 CSS3 所增强的版本相比才更有说服力。

1.6.2　使用CSS3 之后

通过创建一个 CSS3 版本的页面，我减少了 9 张图片，并且用纯 CSS 来替代它们（图 1-5）。经过了这些变更后，页面在现代浏览器下看起来和之前的版本几乎完全一样。



图 1-5　使用 CSS3 省去了图中标记所示部分的 9 张图片，结果整个页面和图 1-3 没有两样（另见彩插图 1-5）

我移除了选项卡的 Sprite 图片，在没有使用任何图片的情况下，通过使用 `border-radius` 和 CSS3 渐变效果来重新制作选项卡（图 1-6）。在这些改变之后，选项卡上开始使用真实的文本，从而增强了它的可访问性。



图 1-6 CSS3 制作的导航和原版基本一致

页脚部分的两个选项卡也使用了同样的技术，通过 CSS3 渐变来替代原来的渐变背景图片。CSS3 渐变还替代了“HSRC PROJECT AREAS”和“LATEST NEWS”旁边的背景图片。另外，导航条上也使用 box-shadow 来替代了头部内容底部的微小渐变。

我使用 border-radius 来移除“HSRC PROJECT AREAS”顶部和底部的圆角图片，以及“LATEST NEWS”头部的图片。对于“HSRC PROJECT AREAS”和“LATEST NEWS”两个部分的头部文字，我移除了图片，使用@font-face 来美化真实的文本。我并没有使用和图片一致的文字，而是使用了一种相似的，是因为那种文字基于版权原因不允许被@font-face 引用。在“HSRC PROJECT AREAS”底部的三条边框，我使用了 box-shadow，这个特性能够在不使用图片的情况下创建复杂边框的效果。

这不是 CSS3 在这个页面上唯一能够添加的地方，但是它确实能够快速而轻易地实现，而且不会对不支持的浏览器造成很大的影响。应用 CSS3 后 CSS 文件将会略微变大，但这并不会造成很大影响，因为它们大多数都是在替换旧的长 background 定义。HTML 文件将会是一致的，除了链接到头部的部分会有所改变。

表 1-3 展示了这个页面的性能。即便是使用@font-face 之后增加了两个 HTTP 请求，总数还是有明显下降，因为我移除了九张图片。同时还移除了用来在 IE 6 下支持 PNG 图片 Alpha 透明效果的 JavaScript 文件，已经不再需要它了，因为这儿已经没有任何 Alpha 透明的 PNG 文件。

表1-3 使用CSS3后的页面性能

	Firefox 3.6	IE 8	IE 6
HTTP请求	22	23	24
页面读取时间（单位：秒）	1.1	1	1.5
读取时间减少量	15%	13%	25%

对加载时间的减少率，我是使用之前那个优化成一张图片的版本来进行比较的。平均而言，加载时间在 Firefox 3.6 会减少 15%，在 IE 8 会减少 13%，在 IE 6 会有 25%。当然这只是一个有限的例子，这种优化可以更彻底，通过减少更多的图片，而优化的效果在更大或者更复杂的网站上会更明显，因为它们会比单个页面上多更多的“界面型”图片（这是相对于内容型图片而言，好比照片）。最关键的是，CSS3 能够让页面加载速度明显变快，同时也提高了一点可用性和可访问性。

这会让网站的用户更高兴，而高兴的用户通常也会让网站背后的人们受益。

具有讽刺意味的是，尽快在 IE 6 上并不能感知到我们所做的这些改变，但使用它的用户还是可以从中受益。IE 6 用户同样可以获得更快的页面加载速度，因为 CSS3 效果取代了原来的图片。

但是在 IE 上这个页面看起来具体是怎样的？到底会有多恐怖？你可以自己来了解一下图 1-7。这是 IE 8 上的截图，但是在 IE 6 上我的感觉也是这样的。圆角的内容都变成了正方，也没法再看到渐变效果。它是否看起来够好？我认为没有，它是否看起来很不好？我认为也没有。IE 用户会知道他们错过了一些视觉效果吗？不太可能。即使他们知道了，你认为他们会宁愿选择圆角图片而不是更快的网速吗？

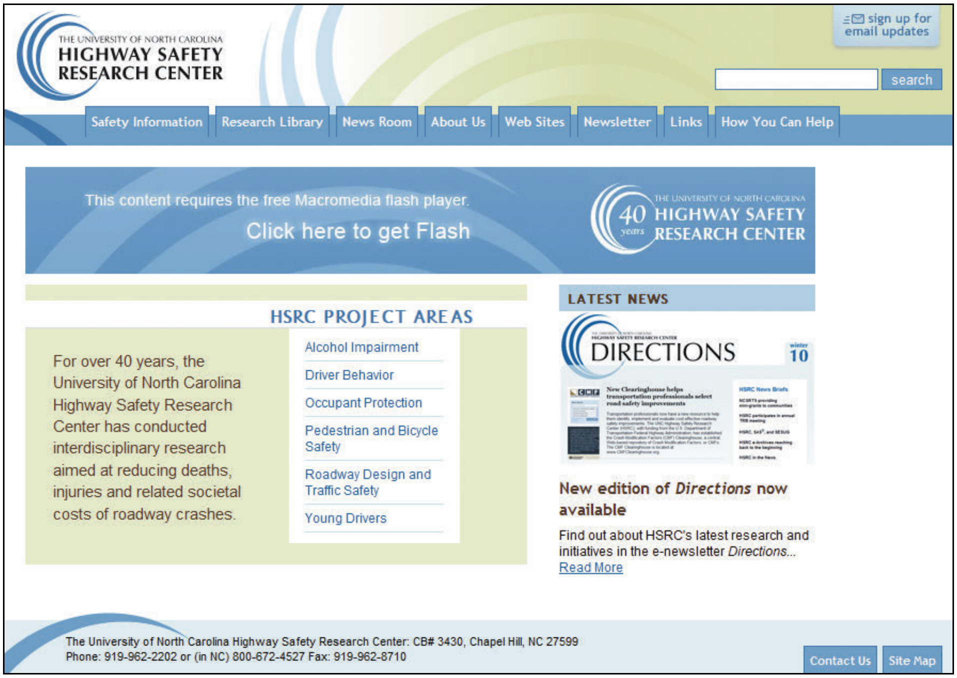


图 1-7　页面在 IE 8 及之前的版本下看起来正常，即便这些浏览器并不支持 CSS3

说明　IE 9 的 beta 版在本书编撰之时已经提供下载，可以显示为所增加的绝大多数 CSS3 效果。当它正式发布时或许能显示更多。

你也可以继续努力让 IE 浏览器上也能够有这些视觉效果，比如添加圆角图片等，但是这么做真的值得吗？你需要做一系列额外的工作，也会消除我们为 IE 用户性能提升所做的努力。这取决于实际的项目，在很多场合这种工作还是很有意义的，在本书中我们就有几次用不同的方法为 IE 添加特定的兼容方案。不是说绝对不能为 IE 添加额外的兼容方案，只是要考虑这个代价而已。

1.7　明智地使用 CSS3

现在你了解到了 CSS3 的使用会带来多大的用处和好处，是否应该马上把它们添加到你的样

式表中去了？当然是的，不过我建议你先了解一些使用 CSS3 的最佳实践。正如 CSS2.1 一样，CSS3 也具有特有的注意事项，比如如何制作样式表从而使它更有组织性，更有效，也更适合未来的发展。你也需要知道如何实现 CSS3 才能不损害到不支持的浏览器，以及如何为特定的浏览器提供不同的兼容方案。

1.7.1　浏览器前缀

当一个浏览器实现了一个新的属性、值或者选择器，而这个特性还不是处于候选推荐标准状态的时候，这属性的前面会添加一个前缀以便于它的渲染引擎识别。举例来说，`-moz-border-radius` 就是基于 Mozilla 的系列浏览器如 Firefox 对于 `border-radius` 属性支持的当前实现。表 1-4 提供了一系列前缀的列表。

表1-4　CSS属性的浏览器前缀

前　　缀	渲染引擎	使用该引擎的浏览器
`-khtml-`	KHTML	Konqueror
`-ms-`	Trident	Internet Explorer
`-moz-`	Mozilla	Firefox、Camino、Flock
`-o-`*	Presto	Opera、Opera Mobile、Opera Mini、Nintendo Wii浏览器
`-webkit-`	Webkit	Safari、Safari on iOS、Chrome、Android浏览器

* 在 Presto 渲染引擎中，对话（Speech）相关的属性前缀是`-xv-`而不是`-o-`。

本书中，我们会关注`-moz-`、`-o-`和`-webkit-`前缀，其他的则因为太不常见而没必要在这里讨论。

1. 为什么它们会存在

浏览器使用前缀来尝试一些新属性、值和选择器，即便它们还没有最终定稿——这是一个好的测试方法，在必要时也可以对它们进行修正或者重新定义。如果浏览器一上来就直接使用标准属性，那它们就会被直接锁定在这个特性的实现上而不易变更。

开发者可能会立即使用无前缀的属性，而且也会一直期望它能够保持同样的表现不再变更。如果浏览器在之后对这个属性做了变更，不管是由于它的实现存在缺陷，或者是由于规范本身发生了变更，所有现存的使用了这个属性的网站都有可能面临出问题的风险。除了会出现这样的锁定问题外，这种方式也有可能强迫其他浏览器和 W3C 去适配它的实现。Eric Meyer 在他的“前缀或者弥补”（www.alistapart.com/articles/perfix-or-posthack）这篇精彩的文章中提到了两个真实的例子，讲述了在过去这种不幸的循环是如何发生的。

即使浏览器不会变更它的实现从而不去影响现有的网站，W3C 如果改变了它的规范定义怎么办？如果其他的浏览器直接使用更新的规范中的新表现该怎么办？现在你有不同的浏览器用不同的方法来实现同样的标准属性。这不正是当年 Netscape 4、Mac IE 5 和 Windows IE 6 并存的那些岁月吗？复杂和不稳定的 Hack，充满了缺陷的实现，甚至这些缺陷与实际所需要实现的属性毫无关系，这些问题不断扩散，因为很多特性都与浏览器规范的属性实际不符。

有前缀的属性会告知开发者这个属性只是实验性质的而且很有可能会变更。它给予浏览器在必要时候进行变更的灵活性，这让浏览器能够更快地发布或者重新定义新的属性。这也让开发者能够更快地有机会尝试新的属性，并且能够参与到在真实环境下测试和修订这些特性的过程。

一旦规范开始稳定下来，浏览器也达到了能够正确实现属性的程度，它就可以去除这个前缀。如果开发者在样式表中同样提供了无前缀的属性的定义——这是很好的向后兼容的方式——那么开发的页面就能够自适应最终定义完成的特性。如果开发者没有这样做，那这些页面也不会受到影响——旧有的前缀属性还是能够正常工作。没有任何一个使用前缀的网站会发生问题。

2. 前缀的问题

浏览器前缀实际上还是存在一些问题的。最主要的抱怨是你需要用几行不同的代码来完成实际上相同的一件事，比如：

```
div {
    -moz-transform: rotate(45deg);
    -o-transform: rotate(45deg);
    -webkit-transform: rotate(45deg);
    transform: rotate(45deg);
}
```

这些重复的代码会增大样式表的文件大小，这样确实挺烦人的。如果使用标准属性的话，你只需要一行描述， 这样就清晰干净多了。许多 CSS 预处理工具事实上可以达到这个效果——你只需要写标准的属性，这些工具会帮你创建浏览器特定的属性。能够这么做的工具包括 Sass（http://sass-lang.com）、LESS（http://lesscss.org）、eCSStender（www.alistapart.com/articles/stop-forking-with-css3）以及其他。但是使用工具来避免书写浏览器前缀也有它的一些不足。如果一种浏览器对某个属性的实现存在缺陷时，你没法在不使用这种浏览器前缀的同时却仍然使用其他浏览器的前缀。如果浏览器的实现有轻微不同的时候，你也没法为这些实现指定不同的值。同样地，添加脚本也会让你的页面变慢。Eric Meyer（www.alistapart.com/aritcles/prefix-or-posthack）认为这种方法可能存在的最大风险在于：

> 如果使用预处理工具来隐藏前缀属性，那代码的写作者会忘记他们在使用实验性的不稳定的特性，因此，会把他们所使用的特性当作一个已经非常稳定的特性。

尽管使用浏览器前缀的重复代码非常笨重，但根据一个标准属性生成不同浏览器前缀属性的方式所造成的不一致性将会更加恼人。而且随着标准的演进，你就能够移除那些浏览器前缀属性从而让样式表更干净，而不用一直修补样式表，仅仅因为一个无前缀的属性突然变成了非标准属性。让我再次引用一下 Eric Meyer 的文章吧，他也提到过这种临时前缀究竟会有多“痛苦”：

> 它有点像疫苗——注射的时候是真的会痛没错，但相对于所能够免疫的疾病来说，那真算不了什么，就这个例子而言，你能够对一个坏的情况免疫，不用持续地做预处理优化和浏览器探测。我们只用一次就避免了一个巨大的危害。正确的使用前缀能够避免在一段长的时间后突然爆发的问题。

提示 当你想要从 CSS 里删除某个前缀时，你可以使用正则表达式来帮你完成这个工作，请参阅 www.venturelab.co.uk/devblog/2010/07/vendor-prefixes-what-happens-next。

前缀的另外一个问题是它们不能通过验证。这不是它本身的问题，验证仅仅是为了帮助我们发现问题。因此如果你知道使用前缀时为什么会有错误提示，只要忽略提示就可以了。但是当这种“良性”的错误混杂在普通的错误中时，很难发现真正的错误。

为了解决包括重复代码和不能通过验证在内的这些问题，有些人把前缀属性和正常的样式表放在不同的文件里。通过这种方法，主要的样式表还能继续通过验证（当然样式表中还可能有其他错误）。但是包括我在内的很多 CSS 开发者并不喜欢这个解决方案。首先，它会增加一个 HTTP 请求，相对于在样式表中增加几字节而言，一个额外的 HTTP 请求肯定对性能的影响更大。其次，它会让你忘记还有个额外的样式表，既然它们相对于正常的特性而言更需要关注，那给予它们更多的关注显然很必要。如果一个浏览器改变了它的某个前缀特性的表现，那你就需要马上更新样式表来保持一致。或者当你想简单地了解一下为什么有些特性会表现成某个特定形式的时候，可能会忘记你还使用了陈旧的前缀样式表，而花费很多时间去跟踪问题。因此非常遗憾地说，相对于为前缀属性保留一个独立的样式表文件而言，由于前缀属性而导致样式表不能通过验证的问题严重程度显然要轻很多。

尽管有这些问题，大部分 CSS 开发者也还都喜欢前缀属性，并且受益于它，正如前面所说，正确使用这些特性就会利大于弊。

3. 使用浏览器特定属性的正确方法

使用浏览器前缀属性的时候，应该总是包含无前缀的属性并将它放在最后。这就可以确保当浏览器可以支持无前缀的属性时，这个属性就能够生效。前缀属性因为位置靠前从而会被靠后的无前缀属性所重载，这才是正确的行为。

举例来说，在 Safari 5 发布之前，Safari 使用`-webkit-border-radius` 属性。它确实曾经很好用，但它的实现从很多方面来看是不正确的（因为 W3C 重定义了规范以后，它就开始不正确了）。Safari 4 及之前的版本不允许单独定义每一个圆角，但是 W3C 的定义说应该可以这样做。它还使用了错误的语法来定义曲线的弧度。

但这是没问题的。你可以保持`-webkit-border-radius` 属性中包含的错误语法，这对其他非 Webkit 浏览器其实是不可见的。而在 Safari 5 中，使用标准属性 `border-radius` 以及正确的语法可以实现圆角特性。甚至都无需修改样式表。标准属性已经在那里了，正在等待使用。

虽然我们说应该总是在最后才包含标准属性，不过可能在有些情况下，我认为应把标准属性留空，只使用浏览器特定的属性版本。如果这个属性的规范定义还在不断变更，建议等它更稳定时再把其标准属性包含进去。当规范最终确定，浏览器开始使用标准属性的时候，这个特定属性很可能会被忽略或者淘汰，像这种情况，就没有必要包含这个属性。

一个很好的例子就是 CSS3 生成的渐变效果。在第 2 章中，你会了解到这个 CSS3 的语法还非常新，Firefox 和 Webkit 在浏览器前缀属性中使用了完全不同的语法。这也许会让你决定根本

就不使用渐变效果，但从另外一方面来说，它只是纯粹的视觉效果，不会对其他浏览器产生负面影响，也许你只是打算把它用在实验性的网站或个人网站上，比如说 iPhone App。即使面临语法变更的风险也想使用渐变特性时，较安全的方法是仅使用前缀版本。当然这样的例子不常见，因为浏览器很少会在语法定义都不太完整稳定时就推出一个特定的前缀版本，即便他们这样做了，你也会继续等待更稳定的版本。

另外一个可选的建议是当你使用某个浏览器前缀版本的时候，最好也加上相关的其他浏览器前缀版本的定义，即使你认为它们现在不会被使用，你也没法否认在将来会有这个可能性。我对此既不赞同也不反对，对我来说，这通常要视情况而定。如果我制作的网站不需要我继续维护，我会考虑加上所有的浏览器前缀属性定义。但如果是一个我会持续维护的网站，我会仅仅加上那些实际需要的浏览器前缀，在以后需要的时候再去添加其他的前缀。这两种方法各有千秋。

不管你选择添加何种前缀属性，在 CSS 中添加注释指出哪个属性是被哪种浏览器所使用永远是个好主意。它其实没有你想象得那么明显。举例来说，这儿就有一组 `border-radius` 属性可能会添加的注释：

```
-moz-border-radius: 20px; /* Firefox */
-webkit-border-radius: 20px; /* Safari 4 and earlier */
border-radius: 20px; /* Opera, Chrome, Safari 5, IE 9 */
```

通过这些注释，当你决定不再支持某种特定的浏览器时，就可以很容易移除那些不再需要的属性了。

说明　在 http://peter.sh/experiments/vendor-prefixed-css-property-overview 页面有一个方便的表格罗列了四大渲染引擎的特有属性。通过它可以对比了解哪些浏览器目前使用带前缀的私有属性，哪些不带前缀，而哪些根本不支持相关属性。

1.7.2　处理不支持的浏览器

这里没有单一的方法能够处理那些不支持你所添加的 CSS3 属性的浏览器。相对地，你选择的方法取决于怎样才最适合网站、用户、客户、你的个人倾向以及 CSS3 技术本身。我们会讨论几个不同的解决方案可以用来处理不支持的浏览器，在本书中，我们也会在恰当时使用它们来解决问题。

1. 接受差异

在很多情况下，处理不支持的浏览器的最佳方法就是直接接受视觉表现上的差异。这就是渐进增强的核心思想，甚至在某些情况下，你根本就毫无选择，因为确实可能某些 CSS3 特性就是没有替代方案。但即使在那些能够找到替代方案的情况下，你也应该问问自己，是否真的有必要这么做。这样做能够真正帮助到你的用户吗？它是否会提高销售额或者订阅量，是否能实现网站目标？在某些情况下，答案是肯定的——那么就选择一个合适的方案来解决差异。但

在很多情况下，答案是否定的，因为 CSS3 效果只是用于美化的，非必要的增强。在有些情况下，添加额外的兼容方案反而会让访客感觉更差，比如在旧式浏览器上添加背景图片会导致页面变慢。

大部分 CSS3 效果在不被支持的情况下根本就不会对浏览器造成负面效果。Twitter 网站（www.twitter.com）就是个很好的例子。Twitter 网站的设计使用 `border-radius` 在多个地方创建圆角效果，以及其他一些 CSS3 特效，这些效果在 IE 8 及以前版本上根本不可见。在最新的非 IE 浏览器上，“What’s happening? ”对话框，就是输入微博的地方，是有圆角效果的，并且在你将鼠标置入输入框的时候周围会有一圈蓝色光晕（图 1-8）。在 IE 8 及之前的版本上，这个输入框只有直角，也没有光晕（图 1-9）。从表现来看，没有任何问题和错误，仅仅是不同而已。这些不同根本不会伤害 IE 用户，因此对于 Twitter 来说，没有任何必要去做额外的工作来保证一致的表现。



图 1-8　Twitter 在 Firefox 下的表现



图 1-9　Twitter 在 IE 下的表现

但是也存在一种可能性，当你不能提供一个可选方案的时候，不支持的浏览器用户会受到损害。举例来说，如果你在制作背景图片的时候使用半透明的颜色如 HSLA 或者 RGBA（这两个声明颜色的方法只有 CSS3 才能支持），不支持 CSS3 的浏览器将会什么颜色都不显示，所以会导致背景完全透明。此时如果文本的颜色和实际的背景色定义不当，整个文本将完全无法阅读（图 1-10）。在这种情况下你无法忽视差异，必须解决这个问题。

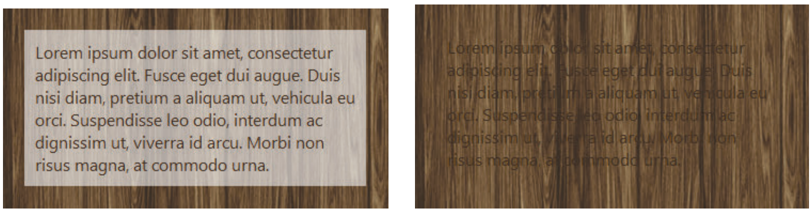


图 1-10　Firefox 下，文本有背景，IE 8 下文本无背景时就不可见

因此，正如之前的 CSS 版本中遇到的那样，我们必须在多种浏览器下进行测试，才能够决定应该选择哪种方案。通常情况下的差异是可以接受的，但也有无法接受的时候。

2. 为一个属性同时提供非 CSS3 和 CSS3 的值

在那些你希望或者不得不提供回滚方案的情形下，可以只为同一条规则提供多于一个的属性值：第一条是为不支持的浏览器准备的；第二条，CSS3 特性值，是为高级浏览器准备的。不支

持的浏览器会忽略那些它们不能理解的属性，而高级浏览器则会用第二条值去重载第一条的值。

举例来说，在之前提到的那个背景色不存在的例子里，你可以先提供一个 HEX 颜色值作为备选方案，然后提供 HSLA 或者 RGBA 的颜色值，例如这样：

```
div {
    background: #CC0000;
    background: hsla(0, 100%, 40%, .5);
}
```

请注意，像这样的方法实际上很难模拟 CSS3 属性所能达到的表现或者行为——备选方案的颜色是实心的，而非半透明。但是它确实提供了一种可以接受的选择，不支持的浏览器的用户至少可以正常使用这个页面了。

3. 使用 Modernizr 检测 CSS3 支持

即便你希望使用不同的值让不支持 CSS3 和支持 CSS3 的浏览器都可用，那也并不总是能在同一规则中指定不同值的，不是每种情况都像上面提到的那个背景例子一样。在某种情况下，两个不同的值会导致崩溃。或者即便它们不崩溃，但你希望提供完全不同的或者更具扩展性的样式给旧式浏览器，而且不希望 CSS3 浏览器加载和解析这些样式。

你可以使用浏览器检测，通过编程，可以检测访客所使用的浏览器，然后为不同的浏览器创建不同的规则。但那是不可靠的，而且有点乱七八糟。一个更好的解决方案是一个名为 Modernizr 的脚本，可以从 www.modernizr.com 下载。它会检测用户的浏览器对 CSS3 和 HTML5 特性的支持程度，然后会对 `html` 元素添加一些类来表明当前浏览器是否支持某些特性，比如“no-multiplebgs”表明浏览器不支持在单一元素上排布多背景图片，“multiplebgs”则表明浏览器是支持的。

当这些类加入时，可以很容易地为每个类编写特定的样式，而且能够确保每个样式只会被支持的浏览器识别。下面的规则就用来为浏览器添加不同的背景颜色或图片，当然这取决于浏览器是否支持这些特性：

```
#wrapper {
   background-color: #ccc;
   background-image: url(one.png), url(two.png),
                     url(three.png);
}
.no-multiplebgs #wrapper {
   background-image: url(alternate.gif);
}
```

第一条规则对所有浏览器可见，不管它们是否支持 JavaScript 或者是否支持 CSS3。不支持多背景的浏览器将会使用背景色，而支持的浏览器将会使用三张背景图片。支持 JavaScript 但不支持多背景的浏览器则会识别出下一条规则，它们会使用规则中提供的单张图片而不是没法支持的三张图片。所以不管浏览器支持哪个级别的 CSS，也不管它是否能够支持 JavaScript，它都能够显示一个正常的背景。

对大部分情况来说，Modernizr 是给不支持的浏览器提供替代性（而不是模拟）样式表的最佳选择。但某些情况下你也可以用它提供对 CSS3 表现和行为的模拟。举例来说，如果你想为盒

容器提供圆角支持，可以在某些浏览器下使用 border-radius，同时在不支持 border-radius 的浏览器下使用背景图片来实现。

```
div {
    -moz-border-radius: 10px;
    -webkit-border-radius: 10px;
    border-radius: 10px;
}
.no-borderradius div {
    background: url(corners.gif);
}
```

说明 实际上 no-borderradius 类的规则定义将比上述代码更为复杂，因为你需要单独为每个边角图片进行定位，有可能需要针对多个 HTML 元素进行定义。我简化上述 CSS 代码的目的就是为了更多关注 Modernizr 发挥的作用。

Modernizr 是一个强有力的资源，我推荐 Frank Ates 的文章“使用 Modernizr 来获得 HTML5 和 CSS3 的优点”（www.alistapart.com/articles/taking-advantage-of-html5-and-css3-with-modernizr），你可以在这里找到更多的例子，也可以了解到如何利用 Modernizr 为不同浏览器定制样式以解决兼容性问题。

4. 使用 JavaScript 模拟 CSS3

我们提供的额外兼容方案内容已经可以为不支持的浏览器提供备选方案了，在很多情况下，只需要提供备选方案而不需要真正模拟。但是如果你希望能够获得完全一致的表现，就需要进行模拟工作。

JavaScript 通常能够使得不支持的浏览器实现和高级浏览器一样的效果，例如现在已经有很多脚本实现圆角效果了。

在本书的每一章，我们都会介绍一些相应的脚本来完成类似的工作，还有一些流行的脚本甚至不会只是模拟某个特性，它们能完成很多的 CSS3 模拟任务。

- IE 7，Dean Edwards 出品（http://code.google.com/p/IE 7-js）。让 CSS3 伪类和属性选择器在 IE 6 到 IE 8 系列浏览器中正常工作。同样模拟了 box-sizing 和 opacity 属性，也包括一些旧版本 IE 浏览器所不支持的 CSS2.1 属性的模拟。
- Selectivizr，Keith Clark 出品（http://selectivizr.com）。让 CSS3 伪类和属性选择器在 IE 6 到 IE 8 系列浏览器中正常工作。它必须和其他 JavaScript 库协同工作。
- cssSandpaper，Zoltan Hawryluk 出品（www.useragentman.com/blog/csssandpaper-a-css3-javascript-library），能够实现 2D 变形、box-shadow、渐变、不透明，RGBA 和 HSLA 在 IE 和其他不支持的浏览器中正常工作。
- PIE，Jason Johnston 出品（http://css3pie.com）。让 border-radius、box-shadow、多背景、background-origin、back-ground-clip 以及线性渐变在 IE 6 到 IE 8 中正常工作。它对 border-image 和 RGBA 也提供了有限支持。

> **说明**　本书出版之后，新的脚本还将继续涌现。想要获取最新 CSS3 的功能模拟脚本，请访问 www.stunningcss3.com/resources。

5. IE 滤镜

另外一个模拟 CSS3 但是不使用 JavaScript 的方法是，使用微软提供的滤镜在样式表中创建不同的视觉效果。这当然只能在 IE 中工作，通常是通过独有的属性滤镜或者`-ms-filter` 属性来实现。具体的属性滤镜语法则依赖于你所使用的特定滤镜，但它们通用的格式是一样的：`filter:progid:DXImageTransform.Microsoft.`**`filtername(sProperties)`**。这里“filtername”是滤镜的名字，而“sProperties”是它的值。在 IE 8 中，这个语法升级成了`-ms-filter` 作为属性名，你也应该用引号来包裹它的属性值。在第 2 章中，可以看到这些滤镜的例子。

完整的可用滤镜列表可以参见这里：http://msdn.microsoft.com/en-us/library/ms532853%28v=VS.85%29.aspx。下面列出的这些特性是可以用来模拟 CSS3 效果的。

- DropShadow、Shadow、Glow 和 Blur 滤镜可以用来模拟 `box-shadow` 和 `text-shadow`。
- Gradient 滤镜可以用来模拟 RGBA、HSLA 和线性渐变效果。
- Alpha 和 BasicImage 滤镜能模拟 `opaticy` 效果。
- Matrix 和 BasicImage 滤镜能模拟 2D 变形效果。

滤镜的优点是它不需要 JavaScript 并且只在 IE 中有效，对其他浏览器没有必要隐藏它们，所以很容易应用。但它们也有如下的一些缺点。

- **长度**。描述一个滤镜将会花费很多字符。如果你在一个样式表中使用很多滤镜，会显著增大这个样式表的大小。为了克服这个，需要将这些样式放在一个单独的文件里面，使用 IE 条件表达式来引入它，关于这个方式我们会在稍后提到。通过这个方式，至少不需要它的浏览器不会下载它的内容。
- **无法通过验证**。包含滤镜的样式表是无法通过验证的。当你明确知道这一点时，它也不再会是一个问题。但如果这让你很烦恼，可以将这些滤镜放在 IE-only 的文件里面从而保证至少你的主样式文件是可以通过验证的。
- **性能**。滤镜会让页面的加载速度变慢，并且消耗大量的内存。
- **粗糙的字体**。滤镜会关闭 Windows 的 ClearType 渲染，因此文本显示将不再是抗锯齿的，这让文本看起来非常粗糙。
- **其他缺陷**。滤镜偶尔还会触发 IE 中的其他缺陷，举例来说，在第 2 章，你会看到一个滤镜是如何让 CSS 产生的内容消失的。

正因为这些问题，我建议只有在真正需要的情况下才使用滤镜。使用时要谨慎，并且做完整的测试。

1.7.3　使用条件表达式过滤IE

通常来说，唯一需要提供特定兼容方案的浏览器就是 IE 6 到 IE 8 系列，所以需要好的方

法可以单独为 IE 提供样式表或者脚本（或者反过来对 IE 隐藏一些东西）。你可能早已经这么做了，单独照顾 IE 可不是 CSS3 提供的新内容。但如果你需要重温一下，这一节会提供几种方法来单独定位 IE。

利用 IE 中 CSS 的缺陷来定位 IE 是最古老的技巧，有很多人还在使用。最流行的也是最有用的 IE Hack 就是星号符（http://css-discuss.incutio.com/wiki/Star_Html_Hack）和下划线（http://wellstyled.com/css-underscore-hack.html）。这些 Hack 的优点是它们能乖乖地待在你的主样式表中，需要改变或者跟踪定位时能很容易地发现它们，它们也不会增加额外的 HTTP 请求。当然，它们中的一些是没法通过验证的，添加它们也会导致所有浏览器的样式表都膨胀。除此之外，除非你能够清晰地记住，否则会较难区分到底是哪个浏览器使用了哪个值，这会让和你同一个团队的开发者很难修改你的代码。

因为这些原因，大部分的 CSS 开发者都使用*条件注释*来定位 IE。条件注释是一个特定的只能被 IE 所识别的 HTML 注释格式。它们是合法的 HTML，所以对其他浏览器不会造成任何损害，所有的非 IE 浏览器都会直接忽略它们，就如忽略其他 HTML 注释一样，条件注释能够让你编写只被 IE 甚至是 IE 的某个特定版本所使用的 HTML。

但我们不是希望 IE 能使用 CSS 特有的样式表，而非 HTML 吗？没错，条件注释能够轻松地通过几种不同的方法来实现这个效果。

说明 用条件注释来加载用于修复 IE 缺陷的脚本也不失为一个好想法。比如上文所列举的那些只针对 IE 的脚本，其他浏览器不会下载这些脚本。

---

1. 提供 IE 独有的样式表

第一种方法是，通过在条件注释中放置一个 `link` 标签或者`@import` 指示符来引入一个 IE 独有的样式表，从而适配所有版本的 IE，就像这样：

```
<!--[if IE]>
<link rel="stylesheet" href="ie_all.css" type="text/css">
<![endif]-->
```

在这个样式表中，你可以在必要的情况下使用 Hack 来适配不同版本的 IE。然而，当 IE 9 出现后，它会更好地支持标准，界时你可能不希望 IE 9 也使用这些 IE Hack。为了避免这种情况，可以这样编写条件注释，保证只有 IE 8 及之前的版本才会使用这个样式表：

```
<!--[if lte IE 8]>
<link rel="stylesheet" href="ie_lte8.css" type="text/css">
<![endif]-->
```

在这里，`lte` 代表了“小于或者等于”。其他可能的值有 `lt` 代表“小于”，`gte` 代表“大于或者等于”，`gt` 代表“大于”。

相对于只使用一个 IE 样式表，另外一个选项是使用多行条件注释为每一个 IE 版本提供一个特定的样式表，就像这样：

```
<!--[if IE 6]>
<link rel="stylesheet" href="ie_6.css" type="text/css">
<![endif]-->
<!--[if IE 7]>
<link rel="stylesheet" href="ie_7.css" type="text/css">
<![endif]-->
<!--[if IE 8]>
<link rel="stylesheet" href="ie_8.css" type="text/css">
<![endif]-->
```

这样就无需在样式表中使用任何 Hack 了，但是这可能会有点难以维护。

条件注释的缺点

虽然条件注释在区分 IE 方面很棒，但它还是有缺点。

- **额外的 HTTP 请求**。每个额外的样式表都会导致一个额外的对主机服务器的请求，这会拖慢页面。
- **对单一对象的规则定义将会分散在多个位置**。这将会增加调试问题的时间（也更容易让人沮丧），得花点时间才能回忆起你在另外一个样式表文件里也对这个对象做了定义。之后如果你需要对样式表做些变更的时候，也容易忘记把这个变更同步到 IE 独有的样式表里面去。
- **阻塞资源文件在 IE 8 下的并行下载**。使用条件注释将会导致 IE 8 阻塞其他资源文件的下载，直到主样式表下载完成。这跟你到底在适配哪个版本的 IE 浏览器无关，也跟你是否用条件注释链入 CSS 文件无关，这个缺陷是 IE 8 与生俱来的，也是影响非常坏的一个，它会明显增加页面的加载时间。唯一的解决方法是在主样式表文件前面增加一个空的条件注释，或者在 `html` 标签下使用条件注释而不是到处使用。之后我们会详细讨论这个解决方案。关于这个缺陷的更多信息，请参见 www.phpied.com/cond-itional-comments-block-downloads。

2. 对 IE 不可见

条件注释也可以用来使内容对 IE 不可见，而不仅仅是为它提供特定内容。这被称为 downlevel-revealed 条件注释（虽然这可不是个非常有帮助的名字）。语法如下：

```
<!--[if !IE]>-->
<link rel="stylesheet" href="not_ie.css" type="text/css">
<!--<![endif]-->
```

`IE` 前面的“!”告诉所有版本的 IE 它们不应该使用注释之中的任何内容，直到结束这个语句的`<![endif]`。

这一次，所有的非 IE 浏览器是可以正确使用这注释之中的内容的，因为这个注释的开始和结束都是独立和完整闭合的注释标记，以下是非 IE 浏览器实际看到的内容：

```
<!-- stuff that doesn't concern me, and now the comment is
over and I should start parsing again -->
<link rel="stylesheet" href="not_ie.css" type="text/css">
<!-- more stuff that doesn't concern me, and now this comment is over -->
```

看到每个注释是如何独立地闭合语句的了吗？没有任何理由浏览器会忽略注释之外的 HTML 内容，IE 之所以忽略它是因为它被设计能够理解这个独特的语法。

你也能针对特定版本的 IE 使用 downlevel-revealed 条件表达式，就如这样：

```
<!--[if !IE 6]>-->
<link rel="stylesheet" href="not_ie6.css" type="text/css">
<!--<![endif]-->
```

说明 欲获取更多有关更复杂、更巧妙的条件注释语法，请参阅 Louis Lazaris 所撰的“Things You Might Not Know About Conditional Comments”一文（www.impressivewebs.com/conditional-comments）。

---

3. 在 html 标签上添加 IE 版本类

另外一个使用条件表达式的方法是不用它链接 IE 独有的样式表文件，而是用它为 html 标签添加一些类来指明当前具体是哪一个 IE 版本的浏览器，然后你就可以在主样式表中为每个类添加特定的属性。这个技术还不常见，但是自从 Paul Irish 在他 2008 年的博文中提到以后就开始流行起来（http://paulirish. com/2008/conditional-stylesheets-vs-css-hacks-answer-neither）。具体的代码如下：

```
<!--[if lt IE 7]> <html class="ie6" lang="en"> <![endif]-->
<!--[if IE 7]>    <html class="ie7" lang="en"> <![endif]-->
<!--[if IE 8]>    <html class="ie8" lang="en"> <![endif]-->
<!--[if IE 9]>    <html class="ie9" lang="en"> <![endif]-->
<!--[if gt IE 9]> <html lang="en"> <![endif]-->
<!--[if !IE]>-->  <html lang="en"> <!--<![endif]-->
```

说明 上述代码中标签之间的空格是为方便阅读而设。如果需要，你可以在页面中删除它们。

为何是 html 标签

如果你愿意，也可以把这些类添加到 body 标签或者某个包裹 div 上去，而不是 html 标签，其实只要是页面上一个能够包裹其他所有元素的标签就可以。

但是 html 标签有它特有的优点：不会阻塞 IE 8 下面的样式表的并行下载（如我在之前所解释过的那样）。如果是 body 或者 div 标签就没法解决这个问题，除非你再额外添加一个空的条件注释到主样式表文件之前。

值得一提的是在 HTML4 和 XHTML1 标准中，类是不能被添加到 html 标签上的。这就是为什么这种技术会让页面没法通过验证，但是在 HTML5 下是可以这样做的，幸好本书中全部使用的都是 HTML5 文档类型！

逐行审查代码的工作量有些大，但做起来非常简单。每一行都简单明了地指明了一个不同的 IE 版本，以及不同的 html 标签。举例来说，如果在 IE 6 下查看，它会作用于条件注释 lt IE 7，说“你好，我是 IE 7 之前的版本，我会使用这里面的内容！”最终就会得到一段 HTML：<html class="IE 6" lang="en">。IE 6 会发现之后的条件注释是不匹配的，所以它就会跳过这些注释，然后一直解释到页面尾部。这个过程在每个 IE 版本的浏览器中都会同样发生，每个浏览器都会获得唯一的 html 标签，这个标签上会添加一个类来指明当前的 IE 版本。

非 IE 浏览器会忽略所有的条件注释，因此它们会忽略前 5 行。第 6 行使用了 downlevel-revealed 条件注释，所以只有非 IE 浏览器才会使用。因此，非 IE 浏览器只会使用最后一行，简单而没有添加类的 html 标签。

当获得了这段 HTML 内容的时候，就可以在你主样式表里为任意一个版本的 IE 创建单独的规则，这无需一个独立的 HTTP 请求，也能保证你的样式都在同一个位置，方便跟踪和调试。你也不需要使用 Hack，也无需担心非 IE 浏览器会看到什么本不应该看到的。举例来说，如果要为不支持 min-height 的 IE 6 浏览器指定一个 height 值，只需要这么做：

```
div { min-height: 100px; }
.ie6 div { height: 100px; }
```

既然添加 IE 独有的规则会增加主样式表的大小，这个增加应该尽可能地最小化，顺利的话，你只需要写一些规避 IE 缺陷的样式，而不用添加很多的杂乱的 Hack。另外，也请记住，就页面性能来说 HTTP 请求比单个请求上增加一点内容更复杂，所以总的来说这个技术对于页面性能和开发维护两方面来说都是有效的。

因为这些优点，我最喜欢最后这个解决方案，这也是本书所选择的解决方案。当然，如果你的偏好并不相同，你完全可以选择自己喜欢的解决方案。

1.8　搞定不予支持的客户或老板

有时成功地使用 CSS3 最主要的阻碍反而不是浏览器的支持度，而是来自于客户或者老板的犹疑。如果你担心这部分人会反对使用 CSS3，可参考以下几种策略。

1.8.1　有些事不必说

让我们先从最简单的“买进”策略开始。坦率地说，我认为可能你的客户或者老板根本不需要知道你在使用 CSS3 这个事实。可以这样思考：如果你雇佣某人去为你建造房子，你没必要也不想知道每件工具和材料的名字，或者这个承包人所使用的建造技术。你也许会关注某些技术细节，但总的来说会更关注总体效果，并且要确保目标是可以用承包人认可的最好方式实现的。

Web 设计在某种程度上也和这个类似。这里有各种各样的技术，但是在实现它之前你的客户或者老板根本无法使用。所以最好的方式就是选择最好的工具并且使用它完成工作。举例来说，也许对于实现一个特定的动画效果来说，讨论究竟使用 CSS3 动画还是 Flash 会很有意义，但确

实没有必要询问客户他们是更愿意用 HSLA 还是 Alpha 透明的 PNG 来实现半透明的背景效果。同样，你也没有必要询问是否要添加一个 `text-shadow` 特性来让文本显得更好看一点。只是用 CSS3 来实现有限的视觉细节时，可以自己决定是否使用，然后直接实现它。

1.8.2 提前教给他们渐进增强的思想

当你还处于一个新项目的竞标阶段时，可以尝试介绍一些关于渐进增强的知识。在开展工作之前，确保客户能够理解渐进增强的基本理念，并且知道这对他的网站会产生怎样的影响。你可能需要展示不同设备和浏览器上的视觉样例（最好是真实的网站）来证明你的观点。介绍一下哪种浏览器你可以提供特别的增强，而哪种浏览器你只会提供基本支持。通过对他现有网站的统计或者是新网站的规划，来找出哪些浏览器对客户最有价值。

当与客户或者老板讨论渐进增强时，没有必要涉及技术细节，倒是可以谈谈在不同的浏览器显示不同的视觉效果是不可避免的发展趋势。为了证明这一点，你可能需要从三个角度来说明这个问题：节省资金，满足用户，以及更好的搜索引擎支持。

可以告知他们，使用渐进增强和 CSS3 技术，能够减少整个项目周期的开发时间和后期维护时间，也能有效降低成本。你可以展示 CSS3 技术是如何通过减少外部资源例如图片数量和文件大小来节省带宽成本的。提醒你的用户为 IE 提供特殊支持是需要消耗成本的，也可以询问这样做能够获得的投资收益率。

你可以谈谈 CSS3 能够提高的用户可用性，而这种提高又是怎样让愉悦的用户在网站停留更久，这将会带来更多的销售额、订阅数或者实现网站需要达到的其他目标。

使用你对 Google 搜索排名逻辑的知识来打动他们，告诉它们 Google 会为更快的网站特别加分，然后说明 CSS3 为何会让网站变得更快。

总而言之，对你的客户或者老板强调渐进增强是符合他们利益的，而且事实就是如此。也许你的观点不能被很快接受，但通过持续的游说，他们就能更好地理解如渐进增强这类 Web 设计开发最佳实践之所以存在的潜在原因。有一天这些实践将会变成主流，这样你就走在了为用户提供最佳价值的潮流最前沿。

1.8.3 控制客户因设计稿而生的过度期望

设计师经常会遇到这样的麻烦事：在给客户展示设计草图的时候，客户可能会期望最终的产品会在所有浏览器上都能具备设计稿中所展示的效果。即使设计稿中的某些特效只能运用于最新的高级浏览器，你可能还是会被迫添加特殊的兼容方案或者一些 Hack 来让所有浏览器都能展示同样的效果。以下有几种方法能够避免这样的结果。

1. 在浏览器中设计

避免不切实际期望的最佳方法是不要使用任何草图，或者不要把它展示给客户。与其用 Photoshop 或者 Fireworks 把你的设计展现成为一幅静态图片，倒不如用 HTML 和 CSS 在最终的媒介即浏览器上实现设计。直接展示给客户可以操作的真实页面。既然展示的就是他自己的浏览

器中所看到的内容，他自然不会要求浏览器所不能支持的特性。

尽管这个方法看起来会花费更多的时间，但是除非客户对你的设计彻底不满意，要求完全重做，否则不出意外的话这个方法不会花费太多时间，事实上，直接使用 HTML 和 CSS 进行设计反而能节约时间。

你需要确定事先已经从客户那里获取了足够多的信息，已经彻底了解了他的期望和品位，不要假设“他看到的时候才知是否喜欢”，直接从他那里获取你所需要的一切。

更多关于不使用图形程序来进行设计的细节

要想获得更多关于在浏览器中进行设计的细节，请参见以下资料。

- “用标记来构建草图”，Meagan Fisher（http://24ways.org/2009/make-your-mockup-in-markup）。
- “Walls Come Tumbling Down presentation slides and transcript”，Andy Clarke （http://forabeautiful-web.com/blog/about/walls_come_tumbling_down_presentation_slides_and_transcript）。

因此，在引入 CSS 工作之前，请使用简单的线框来规划所有的页面结构和布局，这样做的话，就算客户不喜欢你所使用的图片、颜色或者字体，至少所有元素都已经在正确的位置上了，这时再修改就不那么费力了。

事实上，通过编辑 CSS 文件来改变外观通常比修改一个图形草图要容易得多。在图形设计程序中你花费在抗锯齿、行高设定和文本换行的时间，足够在 CSS 中把同样的事情做两次，而且也能更精准地描述这个设计效果在浏览器中的实际表现。这也有助于尽早发现那些只有在真实页面中才会发生的问题，能够尽早地解决， 而不是把它们保留在设计草图中，万一你的客户很喜欢这种问题设计，他们会强迫你花很多时间在真实页面中具体实现这个设计的。

在创意的早期，你还是可以使用图形设计程序来激发灵感，或者是用它们创作那些 CSS 无法胜任的小范围的复杂图像表现。但总的来说，使用真实的样式工具 CSS 构建设计将会让你的工作更简单一些。

除了这些好处，我也明白在浏览器中设计对于大部分 Web 设计师来说是一个巨大的转变，但同时却也是他们的客户特别期望的。我承认我有时也没能这么做。有时你甚至对草图都没有控制权——它们很可能是由其他设计师分别设计的，然后交给你，由在浏览器中实现它。如果处于无法直接在浏览器中设计的情况时，可以参考下面提供的其他避免客户失望的方法。

2. 解释草图的局限性

如果使用传统的设计草图给客户做展示，每个页面只能展示一个视图的情况下，请确保跟客户解释这只是草图，不是最终效果。在展示草图之前，确保让客户知道静态图片从来都不能够完全精确地实现，因为它没法展示浏览器中的各种不同情况、屏幕大小、可能的字体等。要让客户明白他们所看到的每一个视觉细节并不一定能够在所有情况下都可以展示，甚至包括他们自己访问的时候。根据不同浏览器的处理能力的区别，各人看到的实际效果是有轻微差别的，但是你会尽可能地为各个浏览器都提供最佳的体验。

3. 展示可能的情况

如果时间充裕，那就可以考虑为不同的情况都构建对应的草图，例如，可以为主页创建三种不同宽度的草图：480 像素适配移动设备，750 像素适配小显示器，1200 像素适配大显示器。也许你还可以创建一个草图来模拟 IE 8 的表现，比如没有圆角支持和半透明背景的情形。

是的，这样工作量就更大了。但如果它能帮助你避免那些不切实际的期望，同时帮你使用 CSS3 并且获得它的所有好处，那么从长远考虑浪费的时间将会节省回来，而你最终完成的 Web 站点也将会有更好的表现。

第 2 章

气泡对话框

CSS3 最简单有趣的用途之一便是能为页面覆上一层不太起眼的视觉点缀，从而最终化腐朽为神奇。我们将用一些最直观且支持度最广的 CSS3 属性来实现 3D 的气泡对话框（Speech Bubbles），它们可以用来装饰博客评论、突出引用等。

你将学到

我们将实现一些气泡对话框，不需要用到图片，只会用到如下 CSS 特性。

- `word-wrap` 属性，用来容纳溢出的文本；
- `border-radius` 属性，用来实现圆角；
- HSLA 色彩模式，用来创建半透明的背景；
- `linear-gradient` 方法，用来实现背景渐变；
- `box-shadow` 属性，用来实现物件的阴影；
- `text-shadow` 属性，用来实现文字的阴影；
- `transform` 属性，用来旋转物件。

2.1 基础页面

假设你正在着手编写一个博客评论区域的样式。为了兼容那些不支持 CSS3 的浏览器，在开始 CSS3 的创意构想之前，应该先为页面实现一些基本样式。正如第 1 章所提到的，在把 CSS3 效果作为“渐进增强”的一部分添加到页面中之前，重要的是确保它在那些不支持 CSS3 的浏览器下能够使用，并至少拥有一个像样的界面。

图 2-1 向我们展示了一个拥有基本样式的博客评论区域。每则评论的内容、头像、评论者姓名和日期被整齐地罗列出来，文字经过了修饰，甚至还应用了简单的背景图和边框。这个评论区域没有任何问题，它可用、整洁，而且有吸引力。任何使用旧式浏览器访问该页面的人都不会认为自己遗漏了什么信息，或者觉得这个页面“破败不堪”。

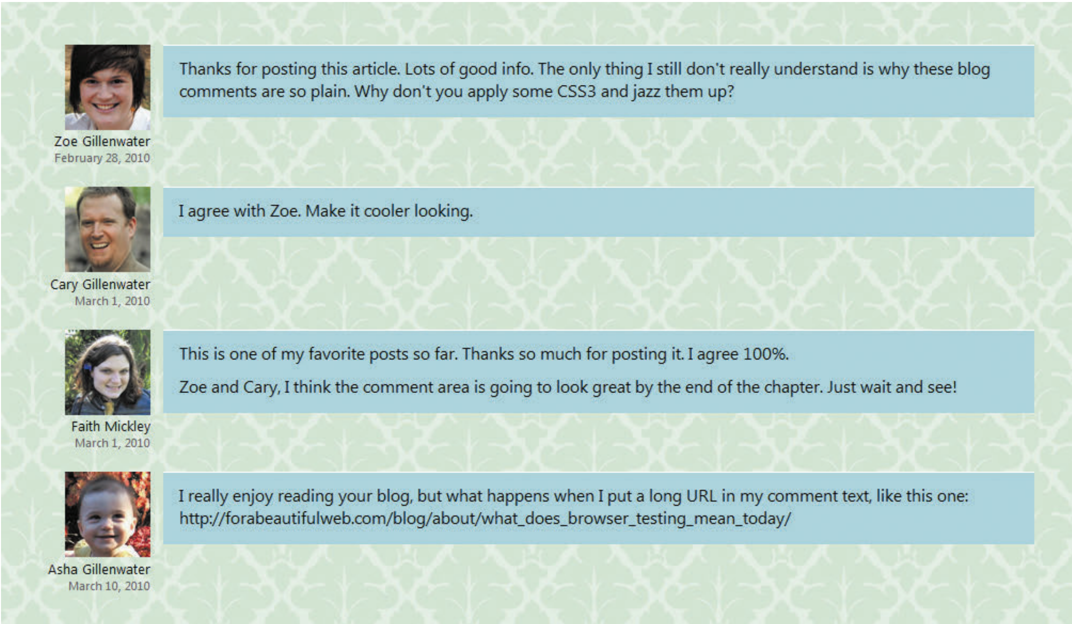


图 2-1 评论区域尚未应用任何 CSS3 特性之前（另见彩插图 2-1）

但是有了 CSS3 的帮助，不需要增加任何图片或是改动页面结构，便可以使页面的展现生动起来。在开始之前，请到 www.stunningcss3.com 下载本章的示例文件，然后用你心仪的代码编辑器打开 speech-bubble_start.html 这个文件。为了便于编辑，相关的 CSS 代码被置于页头的一个 `style` 标签里。

2.2 包裹长文本

是的，我知道我刚才提到要让这块评论区域展现得更生动一些，但在我们正式开始打造气泡对话框的样式之前，请你跟我一起快速地回顾一个困扰大家已久的文本格式问题，解决它只需要用到 CSS3 里最简单的一句代码。

人们常在评论和发帖时附上 URL，而这些 URL 往往会超出文本容器应有的长度（图 2-2）。如果这些 URL 里含有短横线（即“-”），所有主流浏览器可以很好地处理 URL 的折行，然而基于 Webkit 内核的浏览器和 IE 在遇到正斜杠（即“/”）时并不会自动折行，而且所有主流浏览器遇到下划线（即“_”）时均不折行。

在 CSS3 中，终于出现了一个简单方法来强制浏览器进行词内折行以防止文本溢出，你所需要做的只是给 `word-wrap` 属性赋值为 `break-word`。

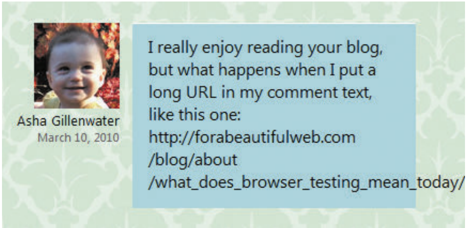


图 2-2　过长的 URL 通常会溢出其父容器，特别是包含下划线的 URL

说明　让我告诉大家一些好消息：IE 早在 5.5 版本中就支持了 word-wrap 属性，所以事实上该属性是由微软创造，随后才被 W3C 接纳的。

---

word-wrap 属性的来龙去脉

word-wrap 属性在“文本”（Text）模块中定义，你可以在 www.w3.org/tr/css3-text 找到它。它控制着文本是否可在词内断开（与 text-wrap 属性不同，它控制词间如何折行）。word-wrap 属性的取值可以是 normal （默认值） 或 break-word。

除了强制断开过长 URL 的需求，你还可以将 word-wrap 应用在如下场景中。

- 防止数据表格因为过宽而溢出或破坏布局，请参阅 www.456bereastreet.com/archive/200704/how_to_prevent_html_tables_from_becoming_too_wide。
- 使 pre 标签里的代码片段折行显示，请参阅 www. longren.org/2006/09/27/wrapping-text-inside-pre-tags。

表2-1　**word-wrap** 属性的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，5.5+	支持，3.5+	支持	支持	支持

在 speech-bubble_start.html 页头的 CSS 里找到 blockquote 规则，然后加入 word-wrap 属性：

```
blockquote {
    margin: 0 0 0 112px;
    padding: 10px 15px 5px 15px;
    border-top: 1px solid #fff;
    background-color: #A6DADC;
    word-wrap: break-word;
}
```

保存修改后的页面并在很狭窄的浏览器窗口中打开。是不是看上去好多了？浏览器会先试着在普通的断开点进行折行，但如果有必要，它将在下划线甚至在某个单词之内进行折行（图 2-3）。不过很显然，在单词内强制折行并不是个好主意，但这种情况总比文本溢出好，况且也只有 URL 才折行，普通文本并没有从单词间断开。

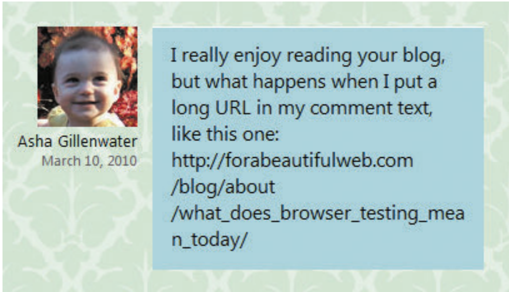


图 2-3　浏览器可能在任意字符之间进行折行

至此已解决了这个小麻烦，接下来把这些评论装扮成气泡对话框。

2.3　无图的图形效果

你可以创建出非常具有图片感的气泡对话框，却无需使用任何真正的图片。不使用图片除了会让你的设计师朋友感到震惊之外还有很多其他好处：可以节省所有用来创建、剪裁和优化图片所花费的时间和精力，还包括当客户不可避免地想要对需求做一些小改动时的返工时间。网站的访客则受益于数据下载量和 HTTP 请求减少带来的速度提高。

说明　第 1 章里介绍了更多有关减少图片数量所带来的好处，还包含一个真实的案例。

2.3.1　圆角的实现

那些棱角分明的直角评论框看起来并不那么像“气泡”，对吧？那就把它打造得更像气泡一些。

圆角是一种简单、常见的视觉效果，但在过去的网页中却是出人意料地难以实现。在绘图软件中制作圆角图片很费时，而用 HTML 和 CSS 去实现一个圆角也是一样。由于 CSS2.1 限制你只能对每个盒容器设置一个背景图，而且用来控制图像定位的 CSS 又十分繁琐，因此时常不得不为每一个圆角图片分别嵌套一层额外的 `div`。这些图片以及臃肿的页面结构和 CSS 增加了每个访客的下载负担，也影响着页面读取的速度。即便是使用脚本动态创建圆角的方式替代手工实现，你还是会增加用户的文件下载量并影响页面的执行效率。而所有的这些麻烦事儿，都源自于那些看似简单的圆角效果！

使用 `border-radius` 来实现（椭）圆形

如果你想要实现的气泡对话框是纯粹的椭圆形而非圆角矩形，那么必须使用椭圆形的边角，而非圆形的。椭圆形圆角框是指每个边角的曲线都被抹平，就像橄榄一样。为了实现椭圆

形圆角，你需要为其指定一对由正斜杠分隔的尺寸值，例如 border-radius: 50px/20px（Safari 3 和 Safari 4 使用非标准的语法：不用斜杠，而用空格），这意味着每个角曲线的横轴长 50 像素，而纵轴只有 20 像素，从而形成一个扁平的椭圆形。你可以为每个边角指定不同的角度，请参阅：http://css-tricks.com/snippets/css/rounded-corners。

要绘制一个正圆，首先请给你的盒容器指定相同的 width 和 height 值，注意使用 em 代替 px 作为尺寸单位来确保它可以根据文字量等比扩大。然后，将每个边角的 border-radius 的尺寸值设置为 width/height 的一半。举例说，假设你的盒容器宽和高都是 10em，则指定 border-radius: 5em。更多示例请参阅：http://blog.creativityden.com/the-hidden-power-of-border-radius-2。

在 CSS3 中实现圆角，简单到只需要给一个 div 指定 border-radius:10px。无需额外的标签结构，无需图片，也不需要使用 JavsScript。

当然，CSS3 还在持续地开发并争取浏览器的支持，因此在实际使用时还稍显繁琐，但真的已经非常简单了。

请在你的页面中修改 blockquote 的相应规则：

```
blockquote {
    margin: 0 0 0 112px;
    padding: 10px 15px 5px 15px;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    border-radius: 20px;
    border-top: 1px solid #fff;
    background-color: #A6DADC;
    word-wrap: break-word;
}
```

border-radius: 20px;这句声明是实现圆角的 W3C 标准语法，它指定所有四个圆角的半径是 20 像素。该语法目前可被 Opera、Chrome、Safari 5 和 IE 9 支持。Firefox 和 Safari 4 及之前的版本则分别使用-moz-border-radius 和-webkit-border-radius 属性。正如我在第 1 章中所述，当某个规范尚在制定之中，而且浏览器认为它还有变更的可能，它们就会使用浏览器引擎特定的前缀。而无前缀的属性（在本例中即 border-radius）必须被置于最末，这样当浏览器确实支持该属性时就会覆盖之前的规则，被覆盖规则很可能依据旧的规范而使用了非标准的行为定义。

说明　使用 border-radius 属性时并不是必须指定边框（border）样式。如果没有边框，则浏览器只为背景区域绘制圆角。

border-radius 属性的来龙去脉

border-radius 属性是“背景和边框模块”（Backgrounds and Borders Module）的一部分，你可以在 www.w3.org/TR/css3-background 找到它。它是用来分别指定圆角半径的四个属性的简写形式，这四个属性依次是 border-top-left-radius、border-top-right-radius、border-bottom-right-radius 和 border-bottom-left-radius。对应每个边角，Mozilla 分别有其私有的非标准属性-moz-border-radius-topleft 等[①]。

你可以在一个 border-radius 属性里为四个角分别指定四个由空格分隔而成的值，或仅用一个值为四个圆角指定相同的半径。在 Safari 4 和 iOS 3 及之前版本的 Safari 下，border-radius 属性只能接受同一个值，而无法在这个简写形式里对四个角进行分别定义。

阅读之前题为“使用 border-radius 来实现（椭）圆形”的补充内容来了解有关椭圆形圆角的语法。你还可以访问 www.owlfolio.org/htmletc/border-radius 和 http://muddledramblings.com/table-of-css3-border-radius-compliance 来获取更多有关 border-radius 属性的语法细节和示例。

除了气泡对话框之外，你还可以将 border-radius 用于如下场景。

- 按钮（Button），请参阅 http://blogfreakz.com/button/css3-button-tutorials 和 http://css-tricks.com/examples/ButtonMaker。
- 标签栏（Tab）。
- 对话框（Dialog box）。
- 圆形标记（Circular badge）。
- 柱状图（Bar chart），请参阅 www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html。
- 表情图（Smiley face），请参阅 http://ryanroberts.co.uk/_dev/experiments/css-border-faces。

表2-2　**border-radius** 属性的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，需要-moz-前缀	支持	支持，5+；4+需要-webkit-前缀	支持

添加上述三行代码之后，除 IE 8 及之前的版本以外的所有浏览器就全部支持圆角了（图 2-4）。这些版本的 IE 则会简单地忽略该属性并依旧显示直角，解决了问题却与用户体验无碍，这是第 1 章所述的“渐进增强”的一个绝佳案例。既然它是一个纯粹的装饰性效果，我认为 IE 用户看不到它并无大碍。但如果你觉得这个方案“确有大碍”，那么请继续向后阅读。

说明　是否看到我多次提到了第 1 章的内容？如果你略过了第 1 章，请立即阅读，里面包含一些重要内容。

① Moziila 的私有属性也有其简写形式：-moz-border-radius。



图 2-4　`border-radius` 属性被应用之后

针对 IE 的兼容方案

如果必须在 IE 8 及之前的版本上实现圆角，你可以从如下脚本中选择一个。

- “PIE”，作者是 Jason Johnston（http://css3pie.com），该脚本将从 CSS 里读取已被定义的 `border-radius` 属性值，并使它们可以在 IE 6 及其之后的版本上工作。它还能让其他若干 CSS3 效果在 IE 上兼容。
- “Curved-Corner”，作者是 Remiz Rahnas（http://code.google.com/p/curved-corner），同样会从你的 CSS 代码里读取 `border-radius` 的值，但它仅支持 4 个角的 `border-radius` 属性值相同的情况。
- “IE-CSS3”，作者是 Nick Fetchak（http://fetchak.com/ie-css3），基于“Curved-Corner”开发而成，对 IE 增加了阴影（drop shadow）效果的兼容。
- “DD_roundies”，作者是 Drew Diller（http://dillerdesign.com/experi- ment/DD_roundies），这个脚本可以让你对四个角进行单独配置，但它并不从 CSS 里读取圆角的定义，你必须针对 IE 手动进行设置。

早在 `border-radius` 属性尚未得到支持之时，除了上述这些针对 IE 的兼容脚本之外，还有不少用来实现圆角的通用型脚本和基于图片的实现技巧，你可以从 www.smileycat.com/miaow/archives/000044.php 和 http://css-discuss.incutio.com/wiki/Rounded_Corners 所介绍的数十种方法中进行筛选。

如果你针对 IE 选择使用脚本或图片的兼容方案，请确保已经限制了脚本来源，或使用了 IE 的条件注释，或直接使用了 Modernizr 来避开其他浏览器，上述方法在第 1 章中均有介绍。根据

此方法，只有 IE 用户会因为使用旧式的圆角实现方案而影响到性能，其他的非 IE 用户则将享受更为高效的纯 CSS 版圆角。你需要权衡的是为 IE 用户实现圆角所花费的额外工作量和造成的性能影响是否值得。

2.3.2 增加气泡的尾巴

有了圆角，现在每个评论框看起来就更像个气泡了，但缺少箭头的气泡对话框终究是不完整的，通常把这个指向发言者的箭头称为“尾巴”。我们可以不依赖任何图片就制作出这条尾巴，事实上，甚至不需用到 CSS3，只需使用 CSS2 的属性和选择器即可实现。

1. 利用边框做出三角形

制作尾巴的必须素材就是一个三角形，而实现它你只需要用到纯 CSS 的普通边框。当一个盒容器的两条边框在边角处相交时，浏览器就会在交点处按某个角度绘制接合线（图 2-5）。如果你将这个盒容器的 `width` 和 `height` 的设置为 0，并为每条边框设置一个较粗的 `width` 值和彼此不同的颜色，最终会得到四个被拼接到一起的三角形，它们分别指向不同的方向（图 2-6）。

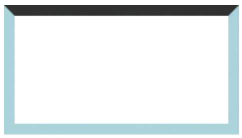


图 2-5 将上边框设为不同的颜色之后，你可以看到边框按某个角度两两接合在一起



图 2-6 当一个盒容器的宽和高为 0，则每条边框都变成一个三角形

以下是实现图 2-6 所呈现的效果需要用到的 HTML 和 CSS 代码片段：

```
<div class="triangles"></div>

.triangles {
   border-color: red green blue orange;
   border-style: solid;
   border-width: 20px;
   width: 0;
   height: 0;
}
```

提示 请记得，在 CSS 里当一个属性同时包含四个值时（见代码中的 `border-color` 属性），它们分别对应“上”、“右”、“下”和“左”，呈顺时针方向排列。

那么将上、左和下边框的颜色改为透明会发生什么？是的，仅有右边框可见，从而呈现出一

个指向左侧的三角形（图 2-7）。代码如下：

```
<div class="triangle-left"></div>

.triangle-left {
   border-color: transparent green transparent transparent;
   border-style: solid;
   border-width: 20px;
   width: 0;
   height: 0;
}
```



图 2-7　设置其中三个边框为透明就能呈现出一个三角形

因此，综上所述，实现一个纯 CSS 三角形的要素是：将一个元素的 width 和 height 设置为 0，然后为它设置较粗的边框，并将其中三个边框的颜色设为透明。你还可以为相邻边框设置不同的边框宽度来改变三角形的角度。

2. 生成尾巴

现在我们已经学会如何不依赖图片去绘制一个三角形，接下来就让我们在每则评论的左边增加一个指向左侧的三角形，并使它指向评论者的头像。为了实现这个需求，可以在评论框中嵌套一个 span 或 div 标签，然后将它转换成所需的三角形，但请勿向这个元素里填充内容，我们将使用 CSS 的内容生成特性让其呈现出理想的样子。

内容生成（Generated Content）是 CSS2.1 的一项特性，利用它你可以将内容书写在 CSS 里并让它展现在 HTML 里。当你不希望手动往 HTML 写入某些硬代码时它就变得非常有用，比如用来实现标题前方的章节编号或是链接之后的小图标等。但它不应被用来写入重要内容，因为一旦用户无法访问 CSS 文件就会因此错过该内容。

要使用 CSS 来生成一段内容，你需要用::before 或::after 伪元素（亦被写作:before 和:after）来指定内容将被插入的位置，并用 content 属性设置内容本身。

为何使用双冒号

你可能注意到我书写::before 和::after 伪元素时使用了两个冒号而并非你所常见的一个。这不是拼写错误。CSS3 已经将伪元素（pseudo-element）的前缀更改为双冒号，而伪类（pseudo-class）则保持为单冒号。

只要你愿意，可以继续使用单冒号的版本，它可以兼容。事实上，IE 8 及之前的版本并不支持双冒号的语法，因此在本书中我们将继续沿用单冒号版。你可以在同一个选择器组里同时使用两个版本，比如.caption:before 和 .caption::before { content: "(图)";}。

举例来说，为了给页面上每个图片的标题都增加“(图)”的前缀，你可以使用如下的 CSS：

```
.caption:before {
    content: "(图)";
}
```

上述 CSS 代码会使`<p class="caption">我的猫咪可爱吗？</p>`这段 HTML 渲染成这样：

（图）我的猫咪可爱吗？

回过头再来看我们想要实现的气泡对话框尾巴，这里关注的是 CSS 所生成内容四周的边框，而不是内容本身。因此，让我们来生成一份隐形的内容：一个空格（non-breaking space）[1]。

空格的 HTML 实体写作` `，但在 content 属性值中却无法使用 HTML 实体，需要改用该字符 Unicode 代码点（Code Point）（或引用）的十六进制部分。这事儿听起来很难以理解而且太过“学术”，不过请勿担心，有很多便捷的线上图表可以提供这类信息的查询服务。

这里就有一例：访问 www.digitalmediaminute.com/reference/entity 你将看到 252 个小格子，每个格子里都是一个(X)HTML 实体。请在“Filter entities by keyword”（根据关键词过滤实体）文本框中输入“non-breaking space”，然后 251 个格子将会消失，只留下名为` `的那个实体格子。请将光标悬停到该格子上方（图 2-8），另外两种编码将会显示出来：分别是该字符的数字编码（numerical code）（在本例中即` `）和 Unicode 编码（u00A0）。而你需要的 Unicode 编码的十六进制部分，就是字母“u”后的字符。因此，请将“00A0”复制到剪贴板。

提示 另一个实用的工具名叫 Unicode Code Converter（Unicode 编码转换器），地址是 http://rishida.net/tools/conversion/。输入一个字符或其 HTML 实体名，这个工具就可以将它转换成多种不同的格式，其中就包括十六进制的代码点。

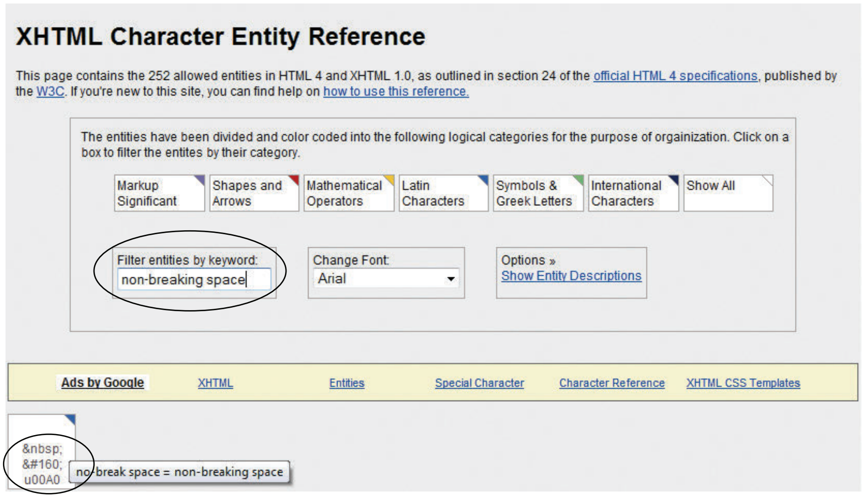


图 2-8 使用 XHTML Character Entity Reference（字符实体参考）来查找不同实体的 Unicode 代码点

---

[1] 它是 nsbp 的全称，可直译为“不折行间隔”；顾名思义，它的插入不会导致内容被浏览器换行。

我们越来越接近终点了，但即使现在有了所需的 Unicode 编码，仍旧不能像这样将它直接赋值给 content 属性：

```
blockquote:after {
   content:"00A0";
}
```

如果这样写，浏览器将非常直截了当地显示“00A0”，而不是我们想要的空格。为了告诉浏览器我们输入的其实是特殊的字符编码，需要对编码进行转义（escape）。如果你是一名程序员，那么应该很熟悉这个术语，但对于其他读者来说，你可以将它理解成必须在编码之前加上一个反斜杠（即“\”）。它会提示浏览器不要把反斜杠之后的字符当做纯文本，而是将它作为一种编码接纳并解析成其他的形式。

有了反斜杠可谓万事俱备，我们终于得以成功插入一个简单的空格了：

```
blockquote:after {
   content:"\00A0";
}
```

说明　Unicode 代码点在书写时通常带有“U+”前缀，而不仅仅是个字母“u”。但无论是哪种形式，能够被 content 属性所用的只能是紧跟在前缀之后的由四位数[①]组成的十六进制编码。

一旦更新了代码，就会发现页面看起来跟修改前一模一样，这当然是因为空格是不可见的。接下来增加边框来让它显示出来，还需要将它的 width 和 height 设置为 0，并使它呈现为块元素（block element）以便将其作为尾巴置于气泡对话框的一侧。

```
blockquote:after {
   content: "\00a0";
   display: block;
   width: 0;
   height: 0;
   border-width: 10px 20px 10px 0;
   border-style: solid;
   border-color: transparent #000 transparent transparent;
}
```

说明　在本例中，使用伪元素:before 和:after 效果都一样，因为我们马上就会让它离开其默认的布局位置。

如果将全部四条边框设置为同样的宽度，我们将会得到一个相当扁平的三角形，就像你在图 2-7 中看到的那样。为了使它看起来更修长一些，我们把上、下边框设置为 10 像素宽，并将左边

① 在十六进制的计数方法中，ABCDEF 六个字母也算作“数字”。

框宽度设置为 0 来让它消失；而右边框——也是将被我们制作成指向左侧的三角形——宽度设置为 20 像素。除了右边框外，其他所有边框均为透明，在这里我先将右边框的颜色临时设置为黑色以便我们能够正确地将其定位（见图 2-9）。

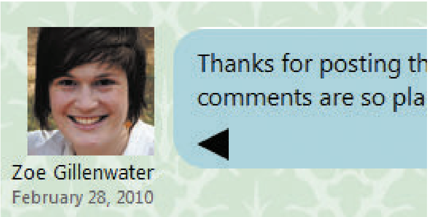


图 2-9　黑色的右边框呈现为一个指向左侧的三角形

这个三角形目前紧跟在 blockquote 内容的后面，并不是一个气泡对话框尾巴理应存在的地方。你可以使用绝对定位法（absolute positioning）来修正其位置。首先，为 blockquote 增加一句 position: relative;样式规则，这将为绝对定位的子元素建立参照点：

```
blockquote {
   position: relative;
   margin: 0 0 0 112px;
   padding: 10px 15px 5px 15px;
   -moz-border-radius: 20px;
   -webkit-border-radius: 20px;
   border-radius: 20px;
   border-top: 1px solid #fff;
   background-color: #A6DADC;
   word-wrap: break-word;
}
```

接下来，调整 top 和 left 的值对生成的内容进行绝对定位：

```
blockquote:after {
   content: "\00a0";
   display: block;
   position: absolute;
   top: 20px;
   left: -20px;
   width: 0;
   height: 0;
   border-width: 10px 20px 10px 0;
   border-style: solid;
   border-color: transparent #000 transparent transparent;
}
```

你可以将 top 设置成任何大于等于 border-radius 的值，以确保它位于圆角的弧线下方，侧对着评论框的直边。为了将三角形拉到最左边，left 值必须为负值，且该值的绝对值刚好等于三角形的宽度。在本例中，三角形的宽度（即右边框的宽度）是 20 像素，因此我们必须将 left 值设置为-20px，它将使三角形恰好衔接着评论框的左缘（见图 2-10）。

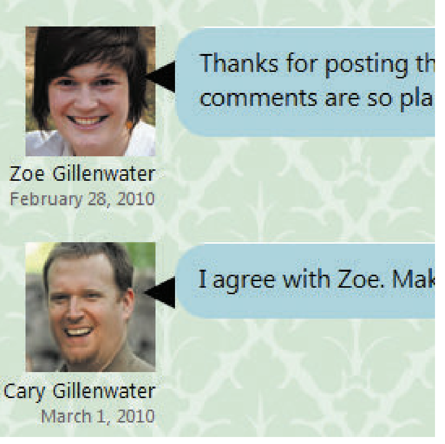


图 2-10　绝对定位法将三角形置于我们想要的地方

有些评论的内容可能太短，以至于三角形过于靠近底边，正如图 2-10 中的第二则评论所显示的那样。要修复这个问题，请为 blockquote 增加一条 min-height:42px;的规则：

```
blockquote {
   position: relative;
   min-height: 42px;
   margin: 0 0 0 112px;
   padding: 10px 15px 5px 15px;
   -moz-border-radius: 20px;
   -webkit-border-radius: 20px;
   border-radius: 20px;
   border-top: 1px solid #fff;
   background-color: #A6DADC;
   word-wrap: break-word;
}
```

现在三角形不再覆盖于 blockquote 之上了，我们可以将它们的颜色统一起来：

```
blockquote:after {
   content: "\00a0";
   display: block;
   position: absolute;
   top: 20px;
   left: -20px;
   width: 0;
   height: 0;
   border-width: 10px 20px 10px 0;
   border-style: solid;
   border-color: transparent #A6DADC transparent
                 transparent;
}
```

本次修改使每个评论框的气泡和尾巴都无缝地结合到了一起（图 2-11）。



图 2-11　每条尾巴都被着色并正确地定位

说明　至此为止的所有修改都在 speech-bubble_1.html 页面里，它在你所下载的本章的示例文件之中。

---

3. 针对 IE 的兼容方案

我们制作的尾巴在 IE 8 及其之后的版本显示正常，但 IE 7 及之前的版本并不支持内容生成的特性，因此这些浏览器的用户将看不到这条尾巴。我想这种情况是可以接受的，这些用户总不会盯着评论框然后自己琢磨着说："等等！为什么这些评论框的旁边就没个小箭头什么的呢？"

要给 IE 7 及之前的版本也加上尾巴，需要手动在 HTML 里为每个评论框额外增加一个元素（比如说一个空白的 span 标签），并想办法把它实现成一个三角形。

2.3.3　利用RGBA或HSLA实现半透明背景

对于气泡对话框的外观，我们已经实现了圆角和小尾巴，能做的事已经不多了，但如果能附加一些图片细节，为它增添更多的视觉层次感和饱满度，或许效果会更好。

一个增添层次感的好方法是使背景变成半透明（Semitransparent 或 Alpha transparency）。通过显露一点页面背景的方式，你就能使界面显得更有层次，仿佛半透明的元素是悬浮于背景之上。我认为这种效果与气泡对话框非常相衬，因为"气泡"总是轻飘飘的。

在 CSS3 出现之前，你可以使用一张半透明的 PNG 背景图来实现这个需求。使用背景图的缺点是会增加一次服务器请求，这会使用户加载得更慢一点。如果你还需要兼容 IE 6，那么负载的压力可能会更大一些：因为它需要一个脚本来“认识”半透明的 PNG 图片。另外，由于不能对边框应用背景图，因此你将无法使气泡对话框的尾巴也变成半透明。如果气泡是半透明的，而它的尾巴却不透明，会显得非常奇怪。

1. CSS3 的 RGBA 和 HSLA 色彩模式

在 CSS3 里我们可以使用 RGBA 和 HSLA 两种色彩模式，二者均可以用来在设置颜色的同时也指定其透明度。RGBA 指的是“红色、绿色、蓝色和 Alpha 透明度”（Red-Green- Blue-Alpha），而 HSLA 则代表“色调、饱和度、亮度[①]和 Alpha 透明度”（Hue-Saturation-Lightness- Alpha）。

我们可以用如下任意一种格式来描述被用作对话框底色的暗蓝色。

- HEX[②]：`#A6DADC`
- RGB：`166,218,220`
- RGBA：`166,218,220,1`
- HSL：`182,44%,76%`
- HSLA：`182,44%,76%,1`

它们返回给我们完全一样的颜色，只不过得到的途径不同。这就好比茴香豆的“茴”字写法虽多，但其实都是同一个字。

在 RGBA 模式里，前三个参数分别是红色、绿色和蓝色的强度值，取值从 0～255 或 0%～100%（最常用的应该是 0～225，而非百分数形式）。而在 HSLA 模式里，前三个参数则分别代表色调（0～360）、饱和度（0%～100%）和亮度（0%～100%）。RGBA 和 HSLA 的第四个参数都是透明度，取值从 0（完全透明）到 1（完全不透明）。

说明　CSS3 仍有 `opacity` 属性，但它的作用是使整个元素都半透明，包括前景内容，而不仅仅是背景。

你可以使用绝大多数的绘图软件来取得所选颜色的正确红、绿、蓝值。先用取色器来选取颜色，然后在取色器窗口或颜色对话框中，软件会告诉你该颜色的 HEX 值和 RGB 值（图 2-12）。而获得 HSL 值则需要一些技巧，因为并非所有绘图软件都使用 HSL 色彩模式，比如 Photoshop 就使用 HSB（也称 HSV）模式，它与 HSL 很相似，又不尽相同。如果你是 Mac 的 Snow Leopard 用户，可以试试一款由 Matt Patenaude （http://mattpatenaude.com）开发的免费软件 Colors，你可以用它从屏幕的任何地方取色，并以 HSLA 等格式显示出来。如果你不是 Mac 用户，我建议你选择一款在线的 HSL 取色器和转换工具（详见下页的“在线色彩工具”）。

---

① 三个词叫法众多：Hue 还可叫作色相、色彩，Saturation 也译作彩度、浓度或纯度，Lightness 也称明度。

② 即颜色值的十六进制形式。

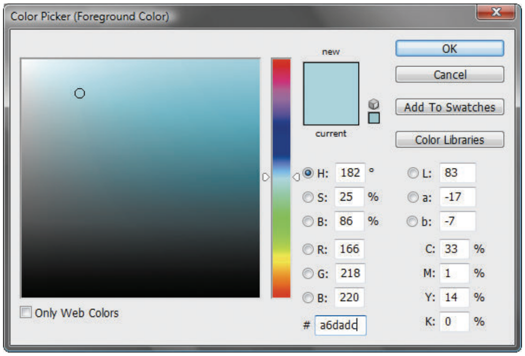


图 2-12　Photoshop 的取色器对话框显示与 HEX 值相对应的 RGB 值

在线色彩工具

Google 一下你就会发现很多基于网页的取色器和转换工具，这里我给大家介绍两款在处理 RGB 和 HSL 值方面特别短小精悍的工具。

- 来自 http://serennu.com/colour/hsltorgb.php 的色彩转换工具可以帮助你在 HEX、RGB 和 HSL 值之间进行转换。
- 而来自 www.workwithcolor.com/doughnut-color-picker-01.htm 的“甜甜圈”取色器则可以方便你取色并转换颜色。这款取色器使用 HSL 色彩模式，但也提供 HEX 和 RGB 的对应值，你也可以手动填写以上任意格式的颜色值进行转换。

使用基于浏览器的取色器来获取 HSL 或 RGB 值将更为简单和快捷。我是 Rainbow 这款 Firefox 扩展（https://addons.mozilla.org/en-US/firefox/addon/14328）的忠实用户。安装它之后可以先指定你想要的颜色格式（图 2-13），接着使用其 Inspector（检视）工具从一个网页里选取颜色时，它会弹出选项，便于将颜色复制到剪贴板（图 2-14）后粘贴到 CSS 代码里。不过需要注意的是，该扩展至今尚不支持 RGBA 和 HSLA 中的“A”参数（即透明度）的显示，因此必须手动添加，不过我想这难不倒你。

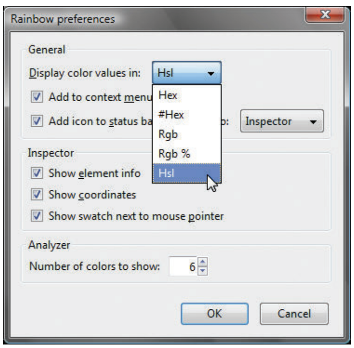


图 2-13　在 Rainbow 的选项面板里，把“Display color values in”选项设置为“Hsl”

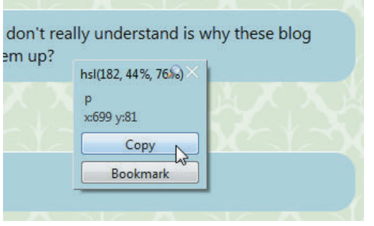


图 2-14　使用 Rainbow 的 Inspector 工具，你可以点击一个颜色并复制它的值

2. RGBA 与 HSLA 之间的抉择

我推荐使用 Rainbow 这款 Firefox 扩展，主要是因为它支持很多扩展不支持的 HSL 模式，而我更推崇使用 HSLA 而非 RGBA。

这方面我应该算是少数派。更多人情愿使用 RGBA 而不是 HSLA，但我想这可能是因为大多数人根本没听说过 HSLA。这很令人遗憾，因为大多数惯用 HSLA 的人发现它更为直观。

碰到一个 RGB/RGBA 值，你很难一眼就看出它所代表的颜色。如果你花上一分钟去研究一个 RGB/RGBA 值，例如 `rgb(166,218,220)`，你可以通过判断三个参数（红、绿、蓝）中哪个值更高的方法来了解它所代表的颜色。但当我面对整张样式表的代码，并试图从中找出某个神秘颜色的定义之处的时候，我想我没有闲心去花时间做上述的研究和判断。而且在我终于发现某个 RGB 值是代表一种蓝绿色调之后，也很难单从红、绿、蓝三个数值就能看出这个颜色究竟有多饱和，或者亮度几何。

HSL/HSLA 模式的色彩速查表

如果你准备使用 HSLA，那么记住少数关键的色调值（或大概了解它们在 0～360 区间中的位置）将很受用。

- 0 或 360 = 红色（red）
- 30 = 橙色（orange）
- 60 = 黄色（yellow）
- 120 = 绿色（green）
- 180 = 青色（cyan）
- 240 = 蓝色（blue）
- 270 = 紫色（purple）
- 300 = 紫红色（magenta）

要获取 HSL/HSLA 格式的黑色（black）值，只需要简单地将亮度设置为 0%；白色则是亮度 100%。对于这两个颜色，色调和饱和度可设置为任意值。

而要获取 HSL/HSLA 格式的灰色（gray）值，则只需将饱和度设置为 0%，此时亮度值控制着灰色的明暗程度，色调的取值则无关紧要。

RGB/RGBA 的另一个问题是，如果你想要对颜色进行调整，例如让它变暗、变浓或是变得

更绿一些的时候，就不得不猜测着去改变每一项的值来得到想要的颜色。Web 设计之中，对页面的不同区块使用相同色系是很常见的事，比如在导航条中对当前选中的 Tab 设置稍亮的颜色。但在 RGB 色彩模式下，同一色系的不同颜色却不一定拥有相似的颜色值。拿我们之前使用过的暗蓝色（166, 218, 220）打个比方，一个只比它稍浓一点的颜色值是（155, 209, 211）。全部三个数字发生了变化，才完成了一次非常微小的饱和度调整。

而使用 HSL/HSLA 来获取一个颜色，你不必同时调整红、绿、蓝三种颜色，只需要将色调值设为一个特定值。你需要做的就是记住 0 和 360 代表着同一种颜色——纯红色，然后按照彩虹的颜色顺序排列，最终重新回到纯红色，就像是一个色盘（图 2-15）。

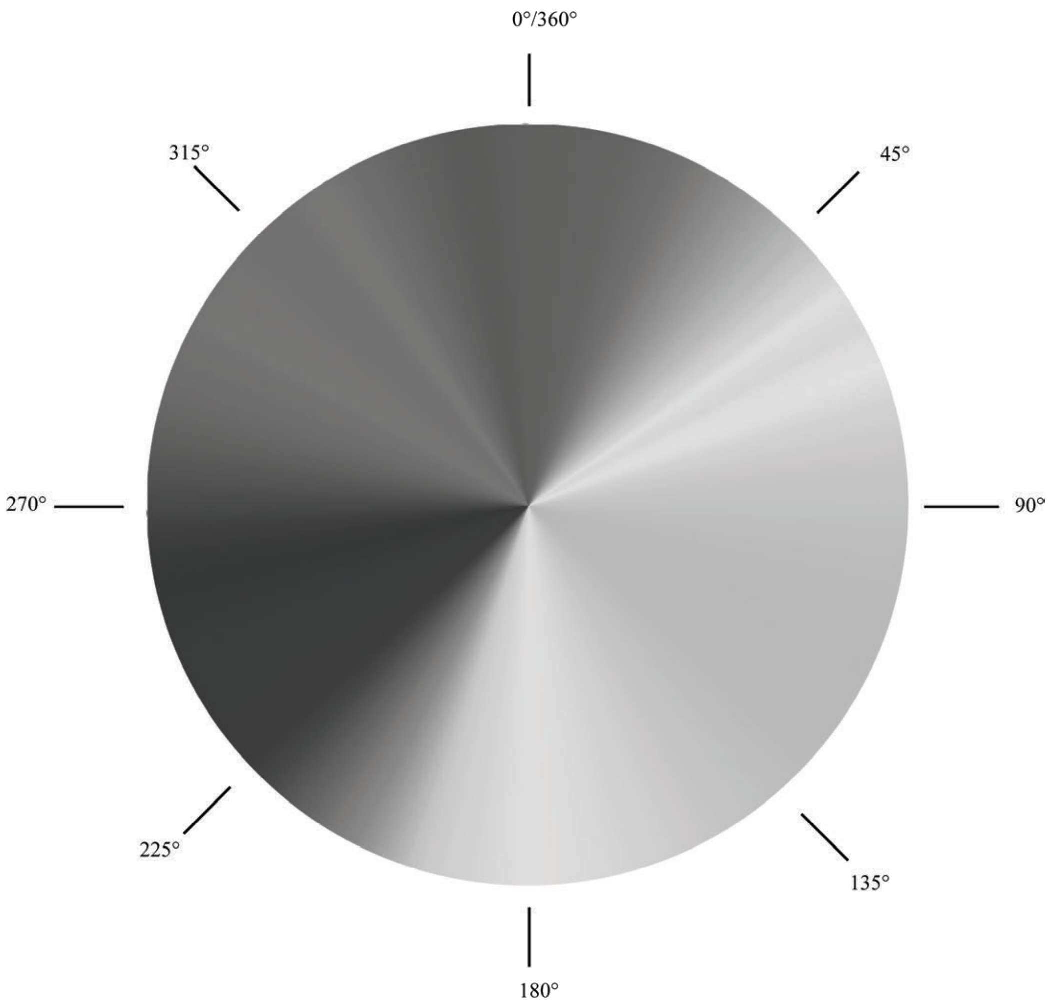


图 2-15　HSL 色彩模式里的 360 个色调（另见彩插图 2-15）

RGBA 和 HSLA 的来龙去脉

RGBA 和 HSLA 是颜色（Color）模块的一部分，你可以在 www.w3.org/TR/css3-color 找到它。可以通过它们来同时设置颜色和透明度。

在 RGBA 模式里，前三个参数分别是红色、绿色和蓝色的强度值，取值为 0～255 或 0%～100%。而在 HSLA 模式里，前三个参数则分别代表：色调（0～360）、饱和度（0%～100%）和亮度（0%～100%）。RGBA 和 HSLA 的第四个参数都是透明度，取值从 0（完全透明）到 1（完全不透明）。

除了实现气泡的背景之外，还可以将 RGBA 和 HSLA 用于下列场景。

- 将阴影投射于背景之上（稍后你将在本章学习如何实现该效果）。
- 按钮或任何其他物件的高光渐变（同上）。
- 为导航栏中当前所选的链接着上比同色调更亮或更暗的颜色。
- 照片上的半透明标题框，请参阅 http://css-tricks.com/text-blocks-over-image 和 www.htmldrive.net/items/show/381/snazzy-hover-effects-using-css3.html。
- 悬浮于内容上方的半透明对话框、模态窗和提示栏（tooltip）。

表2-3　RGBA和HSLA的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持	支持	支持	支持

得到了你想要的色调值之后，如果想让它浓一点或淡一点儿就调整饱和度，想要它更暗一些或亮一些就调整亮度。获取同色系的不同颜色或是对颜色进行微调很简单。开始使用 HSLA 色彩模式后，而且更了解每个色调值意味着什么的时候，当你再遇到样式表中的 HSLA 颜色值时，将一目了然。

总之，RGBA 和 HSLA 的浏览器支持度相同，而且它们也能表示相同的颜色。本书中使用 HSLA 因为我觉得它更直观，但如果你认为 RGBA 更为简单，那么用它也很好。

3. 制作半透明的气泡对话框

弄清楚所有这些模式之后，现在我们可以开始把气泡对话框的背景色由十六进制值改成 HSLA 格式来使它半透明化。

气泡对话框目前的颜色值是#A6DADC。我们可以使用 Rainbow 扩展来找到其对应的 HSLA 值。请用 Firefox 打开页面，并使用 Rainbow 的 Inspector 工具选取气泡对话框的背景色。它将显示其 HSL 值是 hsl(182, 44%, 76%)。复制该值，返回代码编辑器，并用它替换掉目前背景色的十六进制值：

```
blockquote {
   position: relative;
   min-height: 40px;
   margin: 0 0 0 112px;
   padding: 10px 15px 5px 15px;
   -moz-border-radius: 20px;
```

```
    -webkit-border-radius: 20px;
    border-radius: 20px;
    border-top: 1px solid #fff;
    background-color: hsl(182,44%,76%);
    word-wrap: break-word;
}
```

说明　HSL 中用来分隔各个参数逗号之后的空格可有可无，上例中我没有加空格。

如果修改后保存并刷新页面，显示结果将与之前一模一样。你并未修改颜色，只是修改了颜色的格式。

现在我们要修改这种新的格式来使气泡对话框变成半透明。将 `background-color: hsl(182,44%,76%);`修改为 `background-color: hsla(182,44%,76%,.5);`。请务必加上“hsla”中的“a”!

你同样需要修改尾巴的颜色来匹配整体风格。复制 HSLA 值并替换其 `border-color` 的样式声明：

```
blockquote:after {
    content: "\00a0";
    display: block;
    position: absolute;
    top: 20px;
    left: -20px;
    width: 0;
    height: 0;
    border-width: 10px 20px 10px 0;
    border-style: solid;
    border-color: transparent hsla(182,44%,76%,.5)
                  transparent transparent;
}
```

说明　我将透明度的值写作“.5”，但“0.5”也一样。

保存代码并在浏览器中刷新页面。你可以透过气泡对话框隐约看到页面的背景图案，同样地，每条尾巴的尖端也显现出了评论者的头像（图 2-16）。



图 2-16　每个气泡对话框的背景色依旧是暗蓝色，只是半透明化了

4. 针对 IE 的兼容方案

IE 8 及之前的版本不支持 HSLA/RGBA，但有以下备选兼容方案。

- 为这些浏览器指定 HEX、RGB 或 HSL 格式的不透明背景。如果在 HSLA/RGBA 样式之前声明了不透明的背景色，而且两者都使用（或仅有 HSLA/RGBA 使用）`background` 这个简写属性，那么 IE 8 及之前的版本可以忽略 HSLA/RGBA 的样式定义，从而正确使用不透明的版本。但如果不用 `background`，而用 `background-color` 属性来定义 HSLA/RGBA 颜色，IE 7/6 将不会应用不透明的颜色，它们将尝试使用 HSLA/RGBA 颜色，失败后将不会显示任何颜色。在某些页面，文字在没有背景的情况下依旧可读，这种情况尚可接受。但在另外一些时候，没有背景的文字难以辨认，又或是无法应用 `background` 这种简写属性，这就需要针对 IE 7 及之前的版本定义一条专用规则来为其指定不透明的背景色。关于 IE 专有属性的信息请参看第 1 章。
- 将一小张半透明 PNG 图片平铺作为背景图。该方案比起上一种方案来说，优势在于可以真正实现半透明。它适用于 IE 8/7，但不兼容 IE 6 及之前的版本，因为它们不支持 Alpha 透明的 PNG 图片。解决这个问题的方法很多，你可以使用 IE 的 AlphaImageLoader 滤镜，或从众多利用该滤镜来解决透明问题的脚本中选择一个，或给 IE 6 使用 GIF/PNG8 的透明图片，或者干脆为其指定不透明的背景色。但所有的这些方法都意味着额外的工作量，而且对页面的执行效率会产生较大影响——AlphaImageLoader 滤镜运行起来非常慢，而一张图片就是另一次的 HTTP 请求（另外，由于尾巴的部分使用边框来实现，因此上述方法不适用于本例）。除非不需要考虑兼容 IE 6，否则我并不建议使用 PNG 背景图，因为你用不着解决其不支持 Alpha 透明 PNG 图片的问题。
- 使用 Gradient（渐变）滤镜，它自 IE 5.5 起就提供支持了。你可以指定半透明的颜色（当然前提是使用它自身的语法）。将起止色设置为同一种颜色即可避免产生渐变。

提示　你可以使用服务端程序来自动生成 PNG，请参阅 http://leaverou.me/2009/02/bulletproof-cross-browser-rgba-backgrounds。它介绍了一个可以根据你 CSS 中定义的 RGBA 值来生成 PNG 图片的 PHP 脚本。

说明　之前介绍的名为 PIE 的脚本同样可以让 RGBA 兼容 IE，但仅限于一些特定的场景。更多相关信息请参阅 http://css3pie.com/documentation/supported-css3-features/。

我推荐使用第一种或第三种方案。其中第三种方案的实现结果更接近我们对界面的要求，因为背景是半透明的。然而，Gradient 滤镜会对元素文字的抗锯齿效果产生一些奇怪的影响，使它的笔画看起来有点儿不均匀（图 2-17）。你需要在欠美观的文字和更漂亮的背景之间做出抉择。而且，增加这个滤镜会造成 IE 8 上尾巴消失不见（在 IE 7/6 中它未曾显示）。对此现象我无法给出任何解释，它只是 IE 众多奇怪问题中的一个。



图 2-17　IE 8 下 Gradient 滤镜应用前（上）后（下）的对比。下图的背景图虽是半透明的，但文字的抗锯齿效果有些不均匀

这种情况下，我会说："让我们选用滤镜的方案来实现半透明背景吧。"既然对 IE 已放弃了圆角，我不介意再丢掉一条尾巴。

我们可以在 `blockquote` 的样式定义里增加 `filter` 的规则——非 IE 将自动忽略它——但正如我们在第 1 章中所讨论的：最好将 Hack 和兼容代码与普通代码区别开来。为了把滤镜的定义分离出来，我们可以为 IE 单独建立一个样式文件，或使用第 1 章中所介绍的"`html` 标签的条件注释技巧"。这里选择了后者。

打开并找到页面中的 `html` 标签，然后将它更改成如下代码：

```
<!--[if lt IE 7]><html lang="en" class="ie6"><![endif]-->
<!--[if IE 7]><html lang="en" class="ie7"><![endif]-->
<!--[if IE 8]><html lang="en" class="ie8"><![endif]-->
<!--[if IE 9]><html lang="en" class="ie9"><![endif]-->
<!--[if gt IE 9]><html lang="en"><![endif]-->
<!--[if !IE]>--><html lang="en"><!--<![endif]-->
```

提示　如果不想手动输入这些代码（并非责备），请从本章的示例文件中找到 `speech-bubble_final.html` 并打开，将这段代码复制到你自己的文件中。

现在可以开始为 IE 5.5/6/7 创建规则了，由于 IE 8 的滤镜语法与它们不同，我们还需要为 IE 8 单独书写另外一条。首先创建 IE 7 及之前版本的规则：

```
.ie6 blockquote, .ie7 blockquote {
    background: none;
    filter: progid:DXImageTransform.Microsoft.gradient
      (startColorstr=#99A6DADC, endColorstr=#99A6DADC);
    zoom: 1;
}
```

说明　`filter` 规则中的折行是为方便阅读而设。是否保留它悉随尊便，并不影响代码功能。

Gradient 滤镜的渐变起止色被设置成相同的值，这个颜色值看起来有点儿怪，对吗？它们与你所使用的六位数的标准十六进制代码不同。前两位数表示 Alpha 透明度。你可以使用从 `00` 到

FF 的任意十六进制值，其中 00 代表全透明而 FF 则是完全不透明。因此，#99A6DADC 将 Alpha 透明度设置为 99，等同于我们在 HSLA 模式中所使用的 0.6；颜色值则保持 A6DADC，与我们一直使用的暗蓝色相同。

除了加上滤镜之外，我们还在 IE 7 及之前版本的规则里重置了背景色，因为它会覆盖滤镜的颜色。另外，IE 6 及之前的版本还必须触发 blockquote 元素的 hasLayout 状态来使滤镜得以正常工作，我们用 zoom:1;来完成此工作。

说明　理解 hasLayout 很重要。如果你对这个奇怪的“属性”还很陌生，请参阅 Ingo chao 所撰写的文章“On having layout”：www.satzansatz.de/cssd/onhavinglayout.html[①]。

将 RGBA 和 HSLA 值转换成 IE 的 Gradient 滤镜格式

为了在 IE 滤镜里使用与 HSLA 值相同的透明度，你需要用 HSLA 的透明度值（本例中是 0.6）乘以 255，然后将其转换为十六进制值。Robert Nyman 在 http://robertnyman.com/2010/01/11/css-background-transparency-without-affecting-child-elements-through-rgba-and-filters 一文中介绍了如何完成这些步骤。

一个更简便的方法是使用 Michael Bester 编写的工具：“RGBa & HSLa CSS Generator for Internet Explorer”（http://kimili.com/journal/rgba-hsla-css-generator-for-internet-explorer）。输入一个 RGBA 或 HSLA 值，该工具会自动将它转换成对应的 Gradient 滤镜值。

IE 8 无需移除对背景色的定义，因为它会正确地忽略 blockquote 所定义的 HSLA 背景色，也无需触发 hasLayout 状态。但是，它在使用滤镜的语法上稍有不同。请为 IE 8 增加如下规则：

```
.ie8 blockquote {
    -ms-filter: "progid:DXImageTransform.Microsoft.gradient
    (startColorstr=#99A6DADC, endColorstr=#99A6DADC)";
}
```

不同之处在于它使用-ms-filter 而不是 filter，而且该属性的值也必须包含在引号里。新语法更符合 CSS 规范，也更符合其他浏览器设计其私有属性时的习惯。

2.3.4　无图实现色彩渐变

给气泡对话框的背景增加一些渐变效果可进一步增强它的圆润感和立体感。CSS3 让你不依赖图片就能实现渐变，它不仅可以提升你的开发速度，还能减少页面的读取时间，正如之前我们不靠图片实现了圆角一样。CSS 生成的渐变效果还有一个使用图片无法实现的优点——它可以随着外部容器的尺寸变化而伸缩自如，这使得它的用途更为广泛。

但遗憾的是，本书编撰之时 CSS3 的渐变还处在研发阶段，其语法制定还仅在 W3C 的编辑草案里，还未形成一份更完整的工作草案或候选提案。因此，你要明白其语法将比我在书中所介

---

① 国内的 old9 曾翻译过此文，见 http://www.blueidea.com/tech/site/2006/3698.asp。

绍的其他特性更可能发生变动。而且，我认为在有限的范围内使用少量实验性的 CSS 特性是可行的，不支持的浏览器不会因此受到影响，而目前已支持该特性的浏览器也无需担心后续的语法变化会损害浏览体验。最坏（也是最不可能发生）的情况是，它的语法发生了彻底的变化，从而使所有浏览器中的渐变效果都消失了。我认为这个结果是可以接受的。

你可以用它创建出线性渐变（Linear/Straight Gradients）和径向渐变（Radial/Circular/Elliptical Gradients）[①]，本例中我们只关注线性渐变。CSS3 里并没有名为 `gradient` 的属性，可以将 `linear-gradient` 或 `radial-gradient` 函数赋值给任何接受图片的属性，例如 `background-image` 或 `list-style-image` 属性(尽管 Firefox 目前仅支持对 `background-image` 使用渐变)。当你想要使用线性渐变时，必须告诉浏览器起始点、角度和渐变的起止色。你还可以在起止色之间增加一个额外的颜色，并指定每个颜色在渐变中出现的位置。

听起来很简单，但不幸的是，Firefox 和 Webkit（目前仅有的完全支持渐变特性的浏览器）却使用不同的语法，Firefox 符合 W3C 的官方标准，而 Webkit 则使用早期开发时创造的一套完全不同（而且相当复杂）的语法。不仅如此，如果你要实现同一种渐变，具体到每一条规则也有多种变数存在。这着实让人困惑。为了快速入门，我们首先将一个简单的渐变应用到气泡对话框来查看实际效果，然后再深入了解整套语法的细节。

1. Firefox 及 W3C 的语法

Firefox 的语法与正在制定中的 W3C 官方语法相同，它总体上显得更简单，而且易懂易用，所以我们先在 Firefox 上实现渐变效果。

首先，在 `blockquote` 规则的 `background-image` 属性里使用 `-moz-linear-gradient` 函数来为 Firefox 增加一个线性渐变：

```
blockquote {
   position: relative;
   min-height: 40px;
   margin: 0 0 0 112px;
   padding: 10px 15px 5px 15px;
   -moz-border-radius: 20px;
   -webkit-border-radius: 20px;
   border-radius: 20px;
   border-top: 1px solid #fff;
   background-color: hsla(182,44%,76%,.5);
   background-image: -moz-linear-gradient(
          hsla(0,0%,100%,.6),
          hsla(0,0%,100%,0) 30px
          );
          word-wrap: break-word;
}
```

说明 渐变函数中的折行是为方便阅读而设。是否保留它悉随尊便，并不影响代码功能。

① 也称“放射渐变”、“辐射渐变”或“锥形渐变”等。

函数里我们指定了渐变的起始色(hsla(0,0%,100%,.6))、终止色(hsla(0,0%,100%,0))和终止色的位置（30px）。由于并未指定渐变的起始点及其角度，Firefox 将会使用默认值，即从顶部开始垂直向下进行渐变。(如果确实想要指定这两个值，可以在函数开始部分定义。具体语法请参阅“线性渐变的来龙去脉”。)

渐变的起始色是不透明度为 60%的白色，而终止色则是完全透明的白色（不透明值为 0）。将这层半透明的白色覆盖于背景之上会影响其着色 。本例中，它产生的渐变效果是从顶部开始的淡蓝色逐渐向下消失殆尽（图 2-18）。当然我们也可以使用真正的暗蓝色，但在 HSLA 或 RGBA 中使用半透明的白色适用性更广。比如稍后我们会将气泡对话框的背景颜色改成橙色，那么还需要将渐变从淡蓝改成淡橙色。但如果是白色的渐变，它就可以适应任何背景色。坚持使用半透明的白色和黑色是用来调整颜色深浅最聪明的方法。

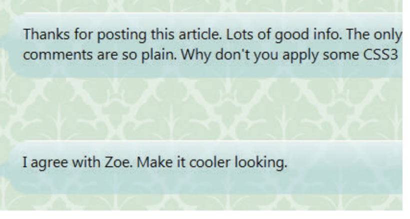


图 2-18　背景上增加一层渐变使气泡对话框看起来更立体了

紧跟着终止色的值之后是一个空格和长度值 30px。这是用来告诉 Firefox 你希望将终止色安排在渐变开始 30 像素以下的位置，接着终止色将填满剩下的区域。既然我们的终止色是全透明的（白色），则给我们呈现出来的就是渐变色仅覆盖了靠近气泡对话框顶端的 30 像素高的部分。

这就是在 Firefox 下实现渐变的方法。通常来说，我还会让你复制(并粘贴)一份 background-image 的规则，然后从这份重复的代码中移除-moz-前缀来增加一份不针对任何特定浏览器的版本。但这一次，由于官方的语法尚在早期的制定中，我认为最好还是等它进入终结阶段再说。我们完成了对 Firefox 的支持，现在继续增加对 Webkit 的实现。

2. Webkit 的语法

针对 Webkit 内核的浏览器，我们对 blockquote 标签增加另一个 background-image 属性，而这一次它包含的是-webkit-gradient 函数：

```
blockquote {
    position: relative;
    min-height: 40px;
    margin: 0 0 0 112px;
    padding: 10px 15px 5px 15px;
    -moz-border-radius: 20px;
    -webkit-border-radius: 20px;
    border-radius: 20px;
    border-top: 1px solid #fff;
    background-color: hsla(169,41%,76%,.5);
    background-image: -moz-linear-gradient(hsla(0,0%,
    ¬100%,.6), hsla(0,0%,100%,0) 30px);
```

```
    background-image: -webkit-gradient(linear,
                    0 0, 0 30,
                   from(hsla(0,0%,100%,.6)),
                   to(hsla(0,0%,100%,0))
                   );
    word-wrap: break-word;
}
```

可见，Webkit 的语法与前一种相比差异巨大而且更为复杂。

径向渐变

我们的例子并未涵盖径向渐变的知识，但从以下文章中可获得更多信息。

- “CSS gradient syntax: comparison of Mozilla and Webkit （Part.2）”，作者 Peter Gasston（www.broken-links.com/2009/11/30/css-gradient-syntax-comparison-of-mozilla-and-webkit-part-2）
- “css gradients in Firefox 3.6”，作者 Alix Franquet（http://hacks.mozilla.org/2009/11/css-gradients-firefox-36）

线性渐变的来龙去脉

渐变函数是图片值（Image Values）草案模块的一部分，你可以在 http://dev.w3.org/csswg/css3-images/#gradients 找到它，但最终的版本将出现于 www.w3.org/tr/css3-images。

你可以在任何接受图片值的属性中使用 `linear-gradient` 或 `radial-gradient` 函数来实现渐变效果。图 2-19 标注了 `linear-gradient` 函数值每个部分的意义。

你不必在使用时包含图 2-19 所示的所有参数。需要记住以下几条重点。

- 在任何语法中，要实现一个渐变只需要指定两种颜色。其余的值都是可选的，如果不填则使用默认值。
- 如果不为颜色指定位置和渐变角度，那么渐变将从顶部延伸到底部。
- 可以在函数的第一个值里指定渐变的起始点，取值可以是关键词（比如 `center`）或数字（比如 `20px` 或 `60%`或 `1em`），可以取负值。如果使用数字，那么第一个值代表水平方向上（`X` 轴）从多远处开始渐变，而第二个值则代表垂直方向上（`Y` 轴）从多深处开始渐变。
- 指定起始点之后，你可以接着指定渐变的角度。它是从水平线到渐变线的逆时针角度。举例来讲，`0deg` 将产生从左至右的水平渐变，`90deg` 则代表从下至上，`180deg` 则从右至左，而 `270deg` 则是从上至下。你还可以使用负值。
- 每个颜色区间都至少包含一个颜色值（对任何语法均适用），还有一个可选的值用来指定该颜色出现的位置。后一个值代表着从起始点到渐变点的距离，起始点不必是盒容器的边界。（图 2-19 中，渐变将从离盒容器顶部 `40px` 的地方开始。）

从本质上来说，既然 `background-image` 属性中的渐变是浏览器生成的图片，可以使用其他 CSS 背景属性来控制它们，就像对待其他图片那样。比方说，可以使用 CSS3 新增的 `background-size` 属性来控制渐变的尺寸，或用 `background-repeat` 来控制是否将其平铺。

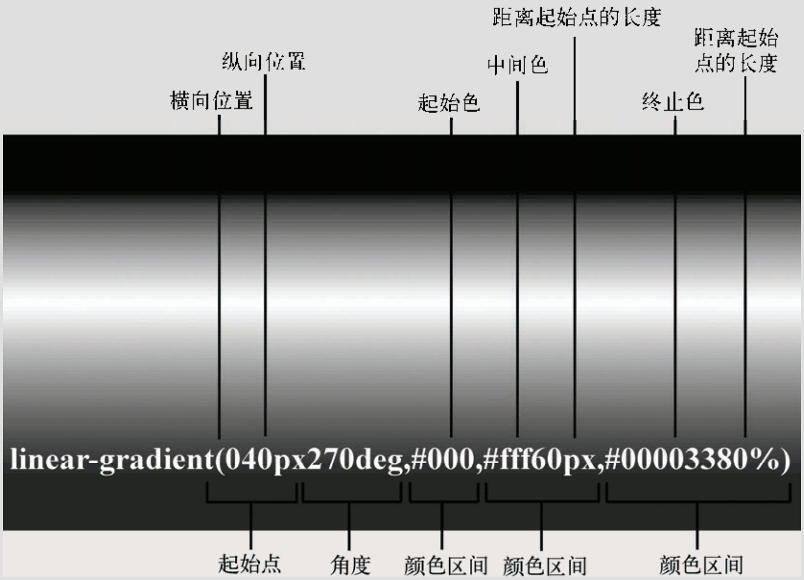


图 2-19　`linear-gradient` 函数每个值所代表的意义及其产生的渐变效果①（另见彩插图 2-19）

除了用 CSS3 渐变来实现高光效果外，你还可以将它用于如下场景。

- 使某些物体的表面看起来像按钮一样呈现弧面，请参阅 www.webdesignerwall.com/tutorials/css3-gradient-buttons 和 http://blog.brandoncash.net/post/525423850/sexy-css-buttons。
- 使一些东西看起来更有光泽，像是金属、玻璃或是 CD。
- 实现倒影效果，请参阅 www.broken-links.com/2010/03/22/create-a-studio-style-backdrop-with-css3。
- 实现晕影（Vignette）效果，使图片或盒容器的边缘像旧照片似的逐渐变暗，请参阅 http://www.broken-links.com/2010/03/22/create-a-studio-style-back-drop-with-css3/。
- 为背景图层叠加一个渐变来实现淡入淡出效果，请参阅 http://atomicrobotdesign.com/blog/htmlcss/make-the-thinkgeek-background-effect-using-css3。
- 等高多列布局，请参阅 http://aext.net/2010/08/css3-sidebar-full-height-background-color。

表2-4　渐变的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	支持，3.6+，需要 `-moz-`前缀	不支持	支持，需要`-webkit-`前缀	支持，需要`-webkit-`前缀

要使用 Webkit 的语法来实现渐变，首先得指定渐变的类型是线性还是径向，并将其作为参

① 根据作者勘正，图中的文字空格间隙过小，现标注如下：`linear-gradient(0 40px 270 deg, #000, #fff 60px, #000033 80%)`。

数传入-webkit-gradient 函数使用，而不是使用 linear-gradient 和 radial- gradient 这两个独立的函数。

然后需要指定水平和垂直两个方向上的起始点位置（本例中是 0 0），紧接着是终止点的位置（本例中是 0 30）。它的值可以是关键词（如 top 或 left）、百分数或像素值。但奇怪的是：如果你使用像素值，Webkit 要求改值但不包含单位值，即"px"。因此在本例中，告诉 Webkit 我们希望渐变的起始点位于盒容器的横向 0 像素和纵向 0 像素处（即左上角的顶点），而终止点则位于横向 0 像素和纵向 30 像素之处。这将使渐变从盒容器的顶部开始并延伸 30 像素，然后将剩下的区域用终止色填充起来，与之前在 Firefox 上的处理方式相同。

指定了起始点和终止点之后，我们还需要指定起止色和终止色。和 Firefox 的语法一样，你可以使用任意颜色格式，但需要注意的是必须把每个颜色包裹在 from()或 to()里。

这句-webkit-gradient 所产生的效果与图 2-18 无异。

Firefox 和 Webkit 之间的语法差别难以记忆。幸运的是，如果你不想，大可不必去记住它。网上有一些很好的渐变生成工具，可用做虚拟的编辑器，创建渐变效果，然后再将所需要的 CSS 输出给你。你要做的只是复制粘贴而已！可以从下列网址找到这些工具：http://gradients.glrzad.com、http://westciv.com/tools/gradients、http://westciv.com/tools/radialgradients 和 http://www.display-inline.fr/projects/css-gradient。

3. 针对其他浏览器的兼容方案

迄今为止，我们所书写的 CSS 只能在 Safari、Chrome 和 Firefox 3.6 及其之后的版本上工作，这使得渐变成为 CSS3 中缺少广泛支持的特性之一。然而，它也是最容易为不支持它的浏览器提供解决方案的特性之一。（如果你想提供兼容方案，恐怕大多数情况下直接让不支持该特性的浏览器显示纯色背景就是一个可以接受的方案。）

使用图片。对于那些不支持该特性的浏览器来说，最显而易见的方案便是使用老办法来实现渐变：创建一个半透明的 PNG 渐变图片，并将其设置为 blockquote 的背景图并横向平铺。请务必将这句 CSS 声明置于包含有-moz-linear-gradient 和-webkit-gradient 函数的 background-image 声明之前，这样才能让支持渐变的浏览器用创造渐变效果的 background-image 属性覆盖指定真实图片的同名规则。

当然，创造并使用图片会抵消你使用 CSS3 来实现渐变所获得的效率提升。Firefox 3.6 不会加载它不需要的图片，但 Safari 和 Chrome 会，尽管它们使用 CSS3 的渐变而且并不需要显示那张图片。这样一来，你在 Firefox 3.6 上成功保留了 CSS3 渐变的效率优势，在 Safari 上却没有。就算你依旧能够享受到由 CSS3 渐变带来的其他优势，但效率却是其中最重要的优势。

有关 Webkit 线性渐变语法的更多细节

在 Webkit 的线性渐变中，如果你想在起止色之间加入其他的颜色，可以使用如下语法：color-stop(50%,#333)。在本例中，50%指定该颜色出现的范围，它也可以是一个 0 至 1 中间的数值。#333 是颜色值，可以是任意的颜色格式。颜色区间可以写在起始色和终止色之间或之后，每个区间用逗号分隔开来。

需要获得更多关于 Webkit 语法的详细信息，请参阅以下文章。

- "Safari CSS Visual Effects Guide: Gradients"（http://developer.apple.com/safari/library/documentation/Internetweb/conceptual/safarivisualeffectsprogguide/gradients/gradients.html）
- "CSS gradient syntax: comparison of Mozilla and Webkit"，作者 Peter Gasston（www.broken-links.com/2009/11/26/css-gradient-syntax-comparison-of-mozilla-and-webkit）
- "CSS gradient syntax: comparison of Mozilla and Webkit (part 2)"，作者 Peter Gasston（www.broken-links.com/2009/11/30/css-gradient-syntax-comparison-of-mozilla-and-webkit-part-2）

正因为 Webkit 内核浏览器的这一点性能问题，我建议要么放弃使用图片做兼容的方案，使那些不支持的浏览器失去渐变效果，要么使用 Modernizr（第 1 章有介绍）对支持渐变特性的浏览器隐藏背景图。当然，如果你正着手解决所有来自于制作和应用渐变图片时所带来的问题，或许你会认为最好是对所有浏览器都使用图片而放弃使用 CSS3 渐变。这个问题没有标准答案，但我的建议是要么仅仅使用 CSS3 来实现渐变，要么根本不用 CSS3 渐变转而使用图片。

提示　另一个模拟渐变效果的方法是嵌入 `box-shadow` 效果，它的支持度比 CSS3 渐变要好（当然仍旧比不上图片）。稍后我们将在本章重温 `box-shadow` 这个属性。

使用脚本。对于 IE 6-8，你可以使用 PIE 脚本（http://css3pie.com/documentation/supported-css3-features）。而对于其他浏览器，请试试 Weston Ruter 的"用 Canvas 模拟 CSS 渐变"（css-gradients-via-canvas）的脚本（http://weston.ruter.net/projects/css-gradients-via-canvas）。它作用于支持 HTML5 Canvas 元素的浏览器，因此可以在 Firefox 2 和 3、Opera 9.64 及其之后的版本中实现渐变。它不支持 IE，但你可以组合使用 IE 的 Gradient 滤镜来实现兼容。这也就引出了下一个方案。

为不同的效果改用不同的颜色。我们之前已经用 IE 的 Gradient 滤镜在 `blockquotes` 创建了一个单色的半透明背景。可以将起始色修改成更淡一些的暗蓝色来模拟 CSS3 渐变。

在所有的 IE 样式规则里，将 Gradient 滤镜中起始色的颜色从 `#99A6DADC` 修改为 `#99E3F4EE`：

```
.ie6 blockquote, .ie7 blockquote {
   background: none;
   filter: progid:DXImageTransform.Microsoft.gradient
      (startColorstr=#99E3F4EE, endColorstr=#99A6DADC);
      zoom: 1;
}
.ie8 blockquote {
   -ms-filter: "progid:DXImageTransform.Microsoft.gradient
      (startColorstr=#99E3F4EE, endColorstr=#99A6DADC)";
}
```

默认情况下 IE 的渐变是从上至下的，因此最终 IE 上的渐变效果将跟 Firefox 和 Webkit 内核

浏览器上的效果相近（图 2-20）。在 IE 中我们无法像 CSS3 渐变那样控制颜色区间的位置，但是滤镜可以用来实现简单的双色线性渐变。它只适用于 IE 8 及之前的版本，IE 9 并不支持滤镜。因此，IE 9 将不显示渐变，但至少它还有半透明的背景色。

提示　更多关于 IE Gradient 滤镜的信息，请参阅 http://msdn.microsoft.com/en-us/library/ms532997(vs.85).aspx。

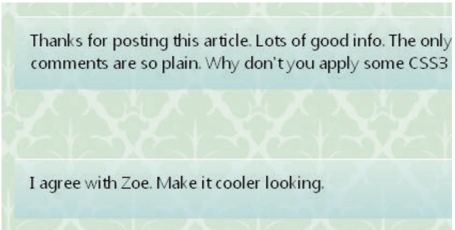


图 2-20　IE 的 Gradient 滤镜可以模拟简单的 CSS3 渐变（图中是 IE 8 的效果）

请注意，如果你为其他浏览器声明过背景图片来做兼容，IE 将使它覆盖 Gradient 滤镜。我们之前已针对 IE 7 及之前的版本移除了背景图，但在 IE 8 的规则里还没有。如果为 `blockquote` 增加了背景图（比如之前选择用图片来实现渐变），请记得在 IE 8 的规则里增加一句 `background: none;`。

说明　至此为止的所有修改都在 `speech-bubble_2.html` 页面里，它在你下载的本章的示例文件之中。

2.3.5　无图实现盒状阴影

为了继续实现界面的立体化，我们可以为每个气泡对话框增加阴影。这一次，同样无需使用图片。

在 CSS3 里，使用 `box-shadow` 属性实现盒容器的阴影。在该属性中，你需要设置阴影相对于盒容器水平及垂直方向上的偏移值、阴影的颜色，还可以选择性地设置模糊半径（Blur Radius）和扩展半径（Spread Radius）。

在 `blockquote` 的样式规则里增加如下三行代码：

```
-moz-box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
-webkit-box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
box-shadow: 1px 1px 2px hsla(0,0%,0%,.3);
```

和之前的 `border-radius` 一样，以上所有代码完成了同一件事，但每一行都针对不同的浏览器；截至本书编撰之时，无前缀的 `box-shadow` 属性将适用于 IE 9 和 Opera。

每个属性值的第一个 1px 表示相对于盒容器的水平偏移，它告诉浏览器将阴影从盒容器的边界向右移动 1 像素。第二个值 1px 则是垂直偏移，它将阴影向下移动 1 像素。你可以使用负值来使阴影向左和向上移动。

第三个值 2px 表示模糊半径，它指定有多少像素的阴影将被拉伸。一个很大的值将使阴影更模糊更朦胧，而取值为 0 将使它有锐利的边界。

说明 box-shadow 属性中的数字值必须按照规定的顺序书写，但颜色值不必写在末尾。如果你愿意，也可以将它写在最前面。

第四个值是阴影的颜色，在本例中是拥有 30%不透明度的黑色。你可以在 box-shadow 中使用任意格式的颜色值，但 HSLA 和 RGBA 将是你最佳的选择——因为它们可以实现半透明色。半透明度对阴影而言显得相当受用，因为你也希望透过阴影能看到一些背景。打个比方，如果将阴影设置为淡灰色，然后将页面背景改为暗海军蓝，最终的效果是一个淡灰色的阴影悬浮于海军蓝的背景之上。其实你真正想看到的是一个颜色更暗些的海军蓝阴影，就像我们现实生活中所看到的阴影那样。在阴影中使用 HSLA 或 RGBA，并将颜色设置为纯黑（阴影效果）或纯白（发光效果）便可以使阴影显现出下方的背景色或图片，而且并不需要每次都相应地更改阴影本身的颜色，它就好像能自己调整颜色一般。

为 blockquote 增加 box-shadow 属性以后，请保存页面，并在某个不过时的浏览器中预览一下每个气泡对话框右下方的那个墨绿色阴影（图 2-21）。你将会看到阴影也注意到 border-radius 的存在并同样保持着圆角。



图 2-21　box-shadow 属性在每个气泡对话框的下方增加了阴影

我们的阴影效果确实增强了立体感，但当鼠标悬停时我们还可以将气泡对话框向“外”移动一点并以此来进一步增强界面的立体感。气泡对话框离背景越远，阴影必须看起来越大。请添加以下的样式规则来增加鼠标悬停时阴影的偏移值：

```
blockquote:hover {
    top: -2px;
    left: -2px;
    -moz-box-shadow: 3px 3px 2px hsla(0,0%,0%,.3);
    -webkit-box-shadow: 3px 3px 2px hsla(0,0%,0%,.3);
    box-shadow: 3px 3px 2px hsla(0,0%,0%,.3);
}
```

提示 如果需要实现更复杂的阴影效果，可以使用 box-shadow 生成器（http://westciv.com/tools/boxshadows）。不过它不支持扩展半径和内阴影（Inset Shadow）。

代码中 top 和 left 的负数值将使气泡对话框本身产生移动，但阴影的尺寸会从 1px 变化到 3px，这个移动显得更真实（图 2-22）。增大阴影会使气泡对话框显得更靠近用户，而不仅仅是离页面背景更远了。



图 2-22 鼠标悬停于气泡对话框时显示更大的阴影使得它好像跳向你一般

box-shadow 属性的来龙去脉

box-shadow 属性是背景和边框（Backgrounds and Borders）草案模块的一部分，你可以在 http://dev.w3.org/csswg/css3-background/#the-box-shadow 找到它，但最终的版本将出现于

www.w3.org/tr/css3 -background。

在该属性中，你需要设置阴影相对于盒容器水平及垂直方向上的偏移值，还可以设置阴影的颜色。通常你还可以设置模糊半径（默认为 0），并选择性地设置一个扩展半径来控制阴影伸展（正值）或收缩（负值）的宽度。你可以在属性值的开头或末尾增加 `inset` 关键词来使阴影看起来被内嵌于一个盒容器。（扩展半径和 `inset` 关键词不兼容以下浏览器：Safari 4 及之前的版本、iOS 3 及之前的版本里的 Safari、IE 9。）

你可以在同一个盒容器的 `box-shadow` 属性里实现多个阴影，每个阴影值用逗号分隔开来。它们依次层叠在一起，首先被声明的阴影将位于最上方。

除了为盒容器实现基本的阴影之外，你还可以将它用于下列场景。

- 发光效果（不设置位置偏移，并可以选择性地设置一个正数值的扩展半径）。
- 3D 效果的按钮。
- 为盒容器模拟多层边框（使用多组 `box-shadows` 值，并将模糊半径均设为 0 以使阴影有清晰的边界），请参阅 http://weston.ruter.net/2009/06/15/multiple-borders-via-css-box-shadow。
- 模拟渐变效果（使用 `box-shadow` 的 `inset` 关键词来实现内阴影），请参阅 http://girlie-mac.com/blog/2010/02/04/css3-box-shadow-with-inset-values-the-aqua-button-rerevisited、http://graphicpush.com/experiments-with-css3-border-radius-and-box-shadow 和 http://nim-bupani. com/vignettes-with-css3-box-shadows.html。

表2-5 **`box-shadow`** 的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
部分支持，9+	支持，需要`-moz-`前缀，3.5+	支持，10.5+	支持，需要`-webkit-`前缀	支持，需要`-webkit-`前缀

针对 IE 的兼容方案

IE 8 及之前的版本并不支持 `box-shadow` 属性，但正如渐变效果一样，你可以使用 IE 的滤镜来模拟实现。“DropShadow”（盒状阴影）和“Shadow”（阴影）两个滤镜正是为实现阴影而设，而 Glow（发光）滤镜则用于在盒容器四周实现发光效果。坏消息是这些滤镜并不像 CSS3 的 box-shadow 属性那样提供诸多的自定义参数，Chris Casciano 在其名为“CSS3 Box Shadow in Internet Explorer [Blur-Shadow]”的文章（http://placenamehere.com/article/384/CSS3BoxShadowin-InternetExplorerBlurShadow）中阐明了这点。我不认为上述任何滤镜能够实现我们在本例中所需要的效果。

另外在该文中，Chris 展示了一个精妙的技巧，即利用 IE 的“Blur”（模糊）滤镜取代“DropShadow”、“Shadow”和“Glow”滤镜来实现更具真实感的阴影，但该技巧需要在 HTML 里复制一份盒容器的副本，并对该副本进行渲染。另一篇文章（http://dev.opera.com/articles/view/cross-browser-box-shadows）使用了相同的技巧，并且展示了如何使用 Blur 滤镜来实现内阴影。

当你确实必须实现阴影效果时，教程里所需要的那个额外的 HTML 元素副本就成了尚可接受的妥协，对于本例中的气泡对话框来说，我觉得为了获得这点视觉提升就去创建多余的 div 元素并不值得。因此我们将不会使用 Blur 滤镜来实现阴影的兼容，IE 下没有阴影效果同样令人满意。

说明　本章前面所介绍过的 PIE 和 IE-CSS3 脚本同样可以在 IE 下实现阴影。其中 IE-CSS3 脚本的阴影只能是黑色。

2.3.6　无图实现文字阴影

为何只有盒容器享受所有的乐趣，难道文字就不能有自己的阴影吗？令人高兴的是，CSS3 中有个名为 text-shadow 的属性可以做到。

text-shadow 属性能为你提供良好的可访问性与可用性。CSS3 能将一个装饰性的图片替换成几乎相同的图形效果——就像在本章已经看到的那些效果一样，比如说把一个 GIF 圆角图替换成 CSS 生成的圆角，有点儿“以假乱真”的意思。那么从这个方面来讲，text-shadow 属性便是用真实的文本替换掉了文字图片。比方说你希望某段标题文字能有阴影效果。在 text-shadow 出现之前，要实现这个需求，你可能需要绘制一张包含阴影及标题文字本身的图片，并将它显示于你的页面。但面对一张图片，用户却无法通过缩放，更改文字颜色、字体等原本可以针对纯文本而进行的一系列操作来使文字更易识别。然而使用 text-shadow 之后，这些操控权就回到了用户手中。

对纯文本应用 text-shadow 效果还可以使文字和背景区分得更为明显，从而提升可读性。不知你是否看过含有“隐藏式字幕”（Closed Captioning）①的电影？这些字幕的文字几乎都有轻微的阴影或描边装饰，以使它们在各种背景色下显得更为醒目。给网页中的文字加上轻微的阴影同样能使可读性得到大幅提升。

使用纯文本的优势还包括：它能够被搜索到，并可以直接选取进而复制和粘贴，相较于图片或 Flash 影片，更方便你或你的客户对文案进行调整。

当然，和其他许多网页技术一样，当 text-shadow 的使用不得其法时反而会影响可用性。我绝对没有让你给所有的文本都加上阴影效果，因此影响到页面可用性的反面教材不胜枚举。必须始终确保文字在没有阴影效果时同样可以被清晰地辨识。但使用得当的话 text-shadow 绝对可以成为又一款得意工具。

text-shadow 的效果听起来相当不错吧？你该如何利用它呢？

1. 使用阴影来凸显文字

让我们用 text-shadow 给当前鼠标悬停的评论增加更醒目的效果。请在 blockquote:hover 的规则里添加一行代码：

```
text-shadow: 1px 1px 1px hsla(0,0%,100%,.7);
```

---

①“隐藏式字幕”是指仅能通过某些特殊设备才能看到的一种字幕，通常供听障人士使用。

它的语法和 box-shadow 几乎完全相同（唯一的区别就是你不能为 text-shadow 设置扩展半径和 inset 关键词）。它同样有水平偏移、垂直偏移、可选的模糊半径和颜色。不同的是，这一次你不需要增加浏览器前缀：Firefox、Safari、Chrome 和 Opera 全部支持标准的 text-shadow 属性。图 2-23 展示了当鼠标悬停于某个 blockquote 时，其文字显现出的轻微的阴影效果。

提示　与 box-shadow 相同，text-shadow 的颜色使用 HSLA 或 RGBA 的半透明色时能获得最佳效果。



图 2-23　当用户将鼠标悬停于气泡对话框之上时，文字的右下方便出现了白色的阴影

说明　至此为止的所有修改都在 speech-bubble_3.html 页面里，它在你所下载的本章的示例文件之中。

评论者姓名和评论日期是另外两个适合增加阴影效果的地方，因为它们的字号都相当小而且文字下方的背景又很花哨。给它们增加一点细微且清晰的阴影可以使它们更加跃然“纸”上且更方便阅读。

请在现有的.comment-meta 规则里增加如下一行代码：

```
text-shadow: 1px 1px 0 hsla(0,0%,100%,.7);
```

这句代码所产生的效果很不起眼，但依旧值得去做。要是给如此迷你的文字增加很粗的描边不仅显得很怪，而且很可能使它们更难阅读。而我们所添加的细微阴影可以提升对比度，从而使文字更容易辨识一些（图 2-24）。

提示　如果想要实现更复杂的阴影效果，可以使用 text-shadow 生成器（http://westciv.com/tools/shadows）。

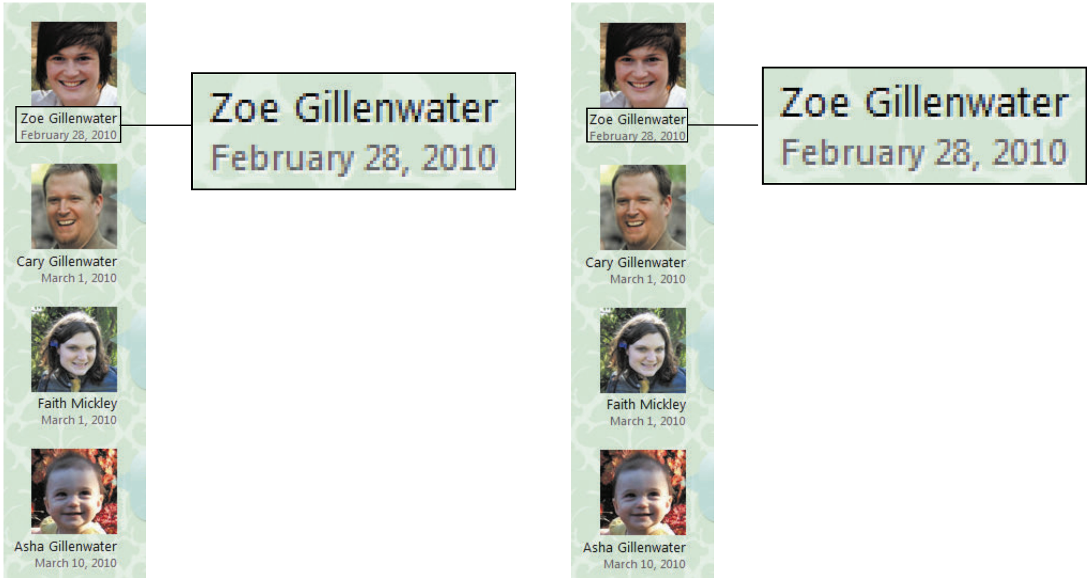


图 2-24 评论者姓名及评论时间下方的实边阴影使文字比修改前显得更为醒目

text-shadow 属性的来龙去脉

text-shadow 是文本模块的一部分，你可以在 www.w3.org/tr/css3-text 找到它。它原属于 CSS 2，但在 2.1 里被移除，现在又回到了 3 之中。

在该属性中，需要设置阴影的颜色以及它在水平和垂直两个方向上相对于文字本身的位置偏移。还可以设置模糊半径，留空则取默认值 0。

可以在同一段文本的 text-shadow 属性里实现多个阴影，每个阴影值用逗号分隔开来。它们依次层叠在一起，首先声明的阴影将位于最上方。

除了用它为文本实现基本的阴影之外，还可以将它用于如下场景。

- 发光文字，请参阅 http://desandro.com/articles/the-new-lens-flare。
- 凸版、镌刻、镂空或浮雕效果的文字（在一边使用淡色阴影，在另一边则使用深色阴影），请参阅 http://sixrevisions.com/css/how-to-create-inset-typography-with-css3。
- 火焰文字（层叠使用黄色、橙色和红色阴影），请参阅 www.css3.info/preview/text-shadow。
- 模糊文字（使用与文字相同颜色的阴影，或干脆使用透明文字），请参阅 http://simurai.com/post/684792689/text-blur。
- 将文字层叠形成 3D 效果（使用多层阴影实现），请参阅 http://css-tricks.com/3d-text-tower。
- 在 http://maettig.com/code/css/text-shadow.html 还可以找到更多 text-shadow 的效果示例，它们之中的一部分更具实用性，但所有的例子都可以用来激发你的创造力。

表2-6　`text-shadow` 的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	支持	支持	支持	支持

2. 针对 IE 的兼容方案

我之前所提到的 IE 的 DropShadow、Shadow 和 Glow 滤镜也可以用来实现文本的阴影，而不仅仅是用于盒容器。为文字实现阴影效果的写法与之前一致，但请确保该元素没有背景颜色或图片。如果它拥有背景，IE 将把阴影渲染给盒容器；反之，阴影才会正确地渲染到文字上。

不幸的是，当上述任意一个滤镜应用于文字时，它会使文字变得粗糙难看，就像我们在使用渐变滤镜时把文字变得不均匀一样（详见图 2-17 和图 2-20），但这一次效果尤其严重。在本章的气泡对话框一例中，我认为这一点确实会影响可读性，更何况增加阴影的目的本身就是为了增强文字的辨识度。因此，在这里我们不会对文字使用任何 IE 的滤镜。如果你仍希望增加滤镜，请从 http://msdn.microsoft.com/en-us/library/ms673539(VS.85).aspx 获取其详细语法。网上还有两个使用 IE 滤镜的 jQuery 插件：http://kilianvalkhof.com/2008/javascript/text-shadow-in-ie-with-jquery 和 www. hintzmann.dk/testcenter/js/jquery/textshadow。

说明　如果你正在使用 Mootools 框架，或许会想要试试它自己的文字阴影脚本，该脚本同时适用于 IE 和非 IE 浏览器：http://pr0digy.com/mootools/text-dropshadows。

2.4　头像的变形效果

我们已经完成了所有针对气泡对话框的修饰，接下来是不是该轮到这些紧靠在对话框旁边的头像图片了？我们当然可以把之前所学到的 CSS3 把戏重耍一遍，但这一次还是试试 CSS3 的新特性变形（Transform）吧。

说明　Firefox 目前还不允许对图片元素使用`-moz-border-radius` 属性，所以很可惜，你无法在 Firefox 下对头像使用圆角。

2.4.1　什么是“变形”

变形是一些效果的集合，每个效果都被称作*变形函数*（Transform Function），它们可以操控盒容器发生旋转、缩放和倾斜等变化。这些效果在之前都需要依赖图片、Flash 或 JavaScript 才能完成。而使用纯 CSS 来完成这些变形则无需加载这些额外的文件，再一次提升了效率，不仅仅是你的开发效率，还有页面本身的执行效率。

transform 属性的来龙去脉

transform 属性是 3D 变形（3D Transforms）模块的一部分，你可以在 www.w3.org/tr/css3-3d-transforms 找到它，也可以在 2D 变形（2D Transforms）模块，www.w3.org/tr/css3-2d-transforms 上找到它。所有的 2D 变形函数也包含于 3D 规范，因此你可以只查阅 3D 规范即可。

变形函数有很多，这里仅列出最重要且支持度最广的那些。

- translate 在指定 X 和 Y 坐标后，能将物体移动到新的位置。取正值是向右、下方移动；反之，取负值则向左、上方移动。示例：translate(20px, -10px)。
- scale 能将物体缩放 X 倍。指定负数值将使物体发生翻转。要使物体变小，请取 0 至 1 之间的小数。如果你指定了一对用逗号分隔开来的数，那么前一个数就是横轴的缩放值，而后一个则用来在纵轴上进行缩放。示例：scale(2.5) 或 scale (1, .5)。
- rotate 可将物体旋转 X 角度（单位是 deg）。正数值按时针方向旋转；负数值则按逆时针旋转。示例：rotate(45deg)。
- skew 能使物体按 X 角度发生倾斜（单位也是 deg）。用逗号分隔的前后两个值分别控制横轴或纵轴的倾斜度，如果仅使用一个值，那么默认纵轴的倾斜度为 0。示例：skew(10deg, 20deg)。

你可以在同一个 transform 属性中使用多个变形函数，并用空格将它们分隔开来。变形将按你书写的顺序进行。

你还可以用 transform-origin 属性来指定变形发生的原点，取值可以是关键词、数值或百分数。默认值是 center。

当你对某个物体进行变形时，周围的其他物体并不会让路给它（类似于相对定位）。该物体将被置于渲染流程（Flow）的首位，然后才开始变形。

除了用它来旋转头像之外，你还可以将它用于如下场景：

- 当鼠标悬停时，增加链接、按钮或表格行的尺寸；
- 当鼠标悬停时，将一个缩略图放大显示；
- 按某种角度摆放的照片（比如说去打造一种照片被随意铺于桌上的场景）；
- 按某种角度排列的便利贴（Sticky-note）；
- 标签云（Tag Cloud）中被不规则排列的标签（请参阅 http://code.almeros.com/how-to-create-a-css3-enabled-tag-cloud）；
- 倾斜的盒容器或图片（增强透视感）；
- 页面顶角处的微型 Banner；
- 边栏文字（比如用来制作流行的博客日期章）（请参阅 http://snook.ca/archives/html_and_css/css-text-rotation）；
- 可打印的折叠卡片（请参阅 http://natbat.net/2009/may/21/pocketbooks）；
- 应用了滑入滑出效果的幻灯片（使用 translate 函数）（请参阅 http://css3.bradshawenterprises.com/slide2/）；

- 当鼠标悬停时显示详情(使用 translate 函数）(请参阅 http://creativefan.com/css3-tutorial-create-card-pockets-how-to)；
- 3D 立方体（使用 2D 的 skew 函数实现）(请参阅 http//depotwebdesigner.com/tutorials/how-to-create-3d-cube-with-css3.html)。

表2-7　2D变形的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	支持，需要-moz-前缀，3.5+	支持，需要-o-前缀，10.5+	支持，需要-webkit-前缀	支持，需要-webkit-前缀

表2-8　3D变形的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	不支持	不支持	支持，需要-webkit-前缀，5+	不支持

和 text-shadow 类似，变形有些时候也可以将某些图片替换成真正的纯文本来提升可用性和可访问性。比如说你想使某个特性列表块或广告文字旋转一些角度。在变形这一特性出现之前，可能需要绘制一张包含文本的图片并将它使用在你的页面里。这种图片的访问体验并不好，因为其文本既不可搜索，也会耗费更多的绘制和修改时间。

很显然你不会对所有的页面文本都使用倾斜和旋转效果，那样做绝对有碍可读性！但少量地用变形替代那些原本需要用图片或 Flash 才能完成的文字效果，能切实提升可访问性。

2.4.2　旋转头像图片

让我们来看看旋转头像所用到的变形代码。请在样式表里增加以下规则：

```
.comment-meta img {
   -moz-transform: rotate(-5deg);
   -o-transform: rotate(-5deg);
   -webkit-transform: rotate(-5deg);
   transform: rotate(-5deg);
}
```

无前缀的那个 transform 属性尚未得到任何浏览器的支持，我们把它放在最后是为了确保以后兼容。

说明　你可以使用这个小巧的 CSS 变形生成器来替你书写许多不同效果的变形代码，http://westciv.com/tools/transforms。

transform 属性（目前也包括另外三个需要增加浏览器前缀的私有属性）会告诉浏览器你想应用一个变形效果。然后指定想要使用的那个特定的变形函数是 rotate，接着是旋转的度数，

单位是 deg，值为 5。你可以使用正值或负值，其他不同变形函数接受不同类型的值，一个形如 -5deg 的值对于 scale 函数来讲没什么意义，不是吗？但最终它们的使用语法都一样：

```
transform: function(measurements);
```

你还可以使用 transform-origin 属性来指定变形发生的初始点，比如物体的中心点或右上角的顶点。其默认值是绝对中心点，正符合本例的需求，因此不需要再增加一个 transform-origin 属性了。

说明 截至成书之时，Chrome 上头像图片被旋转后的抗锯齿效果非常糟糕，图片变得很难看，而且没有解决方案。你可以选择移除包含-webkit-前缀的那句定义，但这样一来 Safari 上也没有旋转效果了，或者你可以选择忍受这难看的效果。

现在请保存代码并在浏览器中刷新页面。你可以看到头像已经发生了旋转（见图 2-25）。

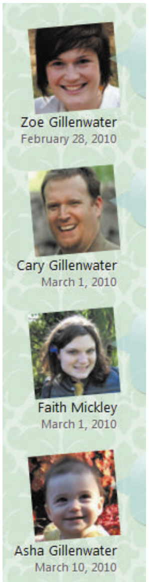


图 2-25 rotate 变形函数旋转了头像图片

针对 IE 的兼容方案

IE 的 Matrix（矩阵）滤镜可以模拟若干 CSS3 变形效果。遗憾的是，你必须进行一系列复杂的矩阵计算才能找到适用于该滤镜所使用的值。我无法在此细述计算的方法（这样一来恐怕就不是一本 Web 设计类书籍了），如果你对数学不是极端痴迷，我所能做的就是向你推荐一款名为“IE CSS3 变形效果转换器”的线上工具，开发者是 Zoltan Hawryluk 和我。

要使用这款转换器，请在第一步的输入框中输入 rotate(-5deg)和任意宽、高的值（我们不会使用宽和高，但对于这款工具来说它们是必填的）。接着点击“Translate to IE Matrix”（转换

为 IE Matrix 格式）按钮。第二步便会显示两段代码，其中一段适用于支持 CSS3 的浏览器，而另一段则适用于 IE（见图 2-26）。



图 2-26　转换器会将你填入的 CSS3 2D 变形效果代码转换为 IE Matrix 滤镜格式

复制适用于 IE 的那段代码，并粘贴到页面的样式定义部分，顺便移除 CSS 中的注释：

```
#transformedObject {
   -ms-filter: "progid:DXImageTransform.Microsoft.
   ¬Matrix(M11=0.9961946980917454, M12=0.08715574274765871,
   ¬M21=-0.08715574274765871, M22=0.9961946980917454,
   ¬SizingMethod='auto expand')";
    filter: progid:DXImageTransform.Microsoft.Matrix(
            M11=0.9961946980917454,
            M12=0.08715574274765871,
            M21=-0.08715574274765871,
            M22=0.9961946980917454,
            SizingMethod='auto expand');
    margin-left: -11px;
    margin-top: -11px;
}
```

这段样式规则包含了适用于 IE 6 和 IE 7 的 filter 属性，还有针对 IE 8 而设的-ms-filter 属性。需要注意的是，上述代码中-ms-filter 属性的折行符是为了照顾本书宽度而人为添加的，并非由转换器处理而成。请确保你在使用时-ms-filter 的属性值在同一行，否则将不会生效。而 filter 属性是否包含折行则无关紧要。添加 margin（外补丁）值是因为 IE 并不像其他浏览器那样使用统一的变形原点，如不添加则头像图片将会遮盖评论者的姓名。因此，添加外补丁或使用相对定位可以让头像的位置与其浏览器保持一致。

现在我们需要更改选择器以匹配我们希望发生变形的元素，在本例中即.comment-metaimg。我们还要将这条单独的规则拆分成两条：其中一条针对 IE 6 和 IE 7，另一条针对 IE 8。请确保两条规则都包含了外补丁值的定义。

```
.ie6 .comment-meta img, .ie7 .comment-meta img {
    filter: progid:DXImageTransform.Microsoft.Matrix(
             M11=0.9961946980917454,
             M12=0.08715574274765871,
             M21=-0.08715574274765871,
             M22=0.9961946980917454,
             SizingMethod='auto expand');
             margin-left: -11px;
             margin-top: -11px;
}
.ie8 .comment-meta img {
    -ms-filter: "progid:DXImageTransform.Microsoft.Matrix(
    M11=0.9961946980917454, M12=0.08715574274765871,
    ¬M21=-0.08715574274765871, M22=0.9961946980917454,
    ¬SizingMethod='auto expand')";
    margin-left: -11px;
    margin-top: -11px;
}
```

如果你保存页面并在 IE 中预览其效果，会发现它在 IE 7 上工作正常，但在 IE 6 上图片的顶部被截断，而且在 IE 8 上图片会轻微地遮盖评论者的姓名（见图 2-27）。

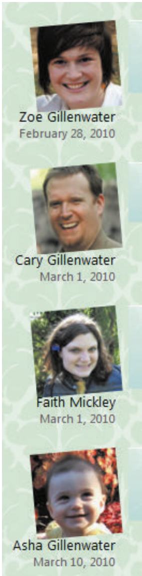


图 2-27　IE 8 下旋转效果正常，但头像的角会遮盖住评论者的姓名

IE 6 的问题是由图片的负数外补丁值引起的，如果使用负值将某个元素移到其盒容器之外，那么 IE 6 不会显示被移出的那个部分。要修正这个问题，请为 IE 6 和 IE 7 的规则添加 `position: relative`。

```
.ie6 .comment-meta img, .ie7 .comment-meta img {
    filter: progid:DXImageTransform.Microsoft.Matrix(
            M11=0.9961946980917454,
            M12=0.08715574274765871,
            M21=-0.08715574274765871,
            M22=0.9961946980917454,
            SizingMethod='auto expand');
    margin-left: -11px;
    margin-top: -11px;
    position: relative;
}
```

而 IE 8 的覆盖问题则是因为我们添加的外补丁值和其他已存在的样式产生了冲突。让我们将负数外补丁值的方案更换为相对定位吧：

```
.ie6 .comment-meta img, .ie7 .comment-meta img {
    filter: progid:DXImageTransform.Microsoft.Matrix(
            M11=0.9961946980917454,
            M12=0.08715574274765871,
            M21=-0.08715574274765871,
            M22=0.9961946980917454,
            SizingMethod='auto expand');
    position: relative;
    top: -5px;
    left: -5px;
}
```

```
.ie8 .comment-meta img {
    -ms-filter: "progid:DXImageTransform.Microsoft.
    ¬Matrix(M11=0.9961946980917454, M12=0.08715574274765871,
    ¬M21=-0.08715574274765871, M22=0.9961946980917454,
    ¬SizingMethod='auto expand')";
    position: relative;
    top: -5px;
    left: -5px;
}
```


说明　请记住，-ms-filter 属性中的折行符均是为了方便排版而添加。请确保你在使用时将它们书写在同一行。

经过这些修改，在 IE 6 至 IE 8 上头像终于可以与其他浏览器保持相同的旋转角度了，而且它们也不会遮盖到下方的评论者姓名或右侧的评论内容了。

除了直接应用 Matrix 滤镜以外，你还可以让脚本在页面执行时默默地帮你解决这个问题。脚本的优势在于它可以帮助你更轻松地创造出基于变形效果的动画。这款同样是由 Zoltan Hawryluk 编写的脚本 cssSandpaper（www.useragentman.com/blog/csssandpaper-a-css3-javascript- library）使若干个变形函数、盒状阴影、渐变、RGBA 和 HSLA 等特性在 IE 上兼容。另一款由 Paul Bakaus 基于 jQuery 编写的简单脚本 Transformie（http://paulbakaus.com/?p=11）同样能解决兼容问题。

2.5　大功告成

至此我们已完成所有针对评论区域的修饰工作，现在可以在一些尚不落伍的浏览器上检查我们的工作成果了，你看到的结果应该和图 2-28 一致。相较于图 2-1 中所展示的那个基础页面，并没有发生本质的改变，但是完成版的页面在视觉上显得更丰富和独特。



图 2-28　Firefox 3.6 上经过 CSS3 改造后的页面（另见彩插图 2-28）

IE 会缺失一些效果，但总体上看起来还是非常接近图 2-28 所展现出来的效果的（图 2-29）。存在一些细小的不同没有什么问题；我们所增加的效果都纯粹用于装饰，更何况 IE 用户其实根本不会知道他们所访问的页面缺少了一些东西。即使完全不添加本章中所有针对 IE 所写的兼容方案（比如滤镜），你的页面在 IE 8 及之前的版本上依旧会如图 2-1 所展示的那样完全可用而且效果诱人。

说明　完成版页面所展示的效果代码都在 speech-bubble_final.html 页面里，它在你所下载的本章的示例文件之中。



图 2-29　页面在 IE 8（左上）、IE 6（右上）及 IE 9 预览版（左下）中并不显示我们所添加的任何 CSS3 效果，但页面看起来依旧漂亮而且可用（另见彩插图 2-29）

第 3 章

纸质笔记

3

第 2 章主要讲述了在不使用图片的情况下如何使用 CSS3 创作图像效果。而在这一章中，我们会大量使用图片，但是通过运用 CSS3 属性，可以用更精简的标记来使用这些图片，并且能实现之前 CSS2.1 无法实现的一些效果。你也会在这一章中了解到如何在不借助 Flash、图片或者脚本的情况下，在页面中使用特定的非 Web 安全字体，甚至在 IE 浏览器上也可以正常使用。总的来说，CCS3 能够让我们通过使用这些图片和字体让一个 Web 页面看起来和一页真实的纸质笔记完全一致。

你将学到

我们将会使用以下这些 CSS3 的属性和概念来创作一页纸质笔记：

- `background-size` 属性可以让背景图片与文本同步缩放；
- 在一个元素上使用多重背景图片；
- `border-image` 属性可以创建图片边框；
- `background-clip` 属性可以让背景图片的位置偏移不再受边框的限制；
- `@font-face` 规则可以在页面中引入特定字体。

3.1　基础页面

在 Web 设计上总有一种流行趋势是模拟真实物体的表现，比如便利贴或者文件夹。如果希望某个页面上的内容看起来像书写在一页真正的纸上，你应该做的第一件事是为这个页面设定一个纸质线条笔记页的背景图像，图 3-1 展示了这个基础的页面。



图 3-1　简单背景图片的页面，尚未应用任何 CSS3 特效

3.2　高级背景图片技术

如果要让图 3-1 中的 Web 页面看起来更像一页真正的纸，只使用简单的背景图像是远远不够的，必须要添加更多的视觉细节，比如撕裂纹理或者咖啡残渍等。当然，不使用 CSS3，你也可以做好这些细节，但是有了 CSS3 的帮助，HTML 代码标签将会更干净，你也能更轻易地实现这些视觉效果。现在就来证明一下吧。

3.2.1　背景图片缩放

如果页面中的文本能够始终和纸质笔记背景图片的线格保持一致，而不是如图 3-1 所示的那样随意交错重叠，那么这个页面的真实感无疑能够大大增强。让我们使用 CSS3 来改善这个问题，你需要以用素值（`px`）设定一个基本的 `font-size` 和 `line-height`，然后调整到可以适配这个背景图片上的线格的行间距。这对大部分用户来说都是没有问题的，但是如果有人主动调整了文本大小，或者用非标准的用户设置覆盖了 `font-size` 的值，这些文本马上就会变得参差不齐。

这是因为文本可以缩放，但是背景图片却不能。

但在 CCS3 引入 background-size 特性之后，这已成为历史。通过使用这一特性，我们可以控制背景图片在水平和垂直两个方向的缩放，也可以控制图片拉伸覆盖背景区域的方式，甚至还可以截取图片。

1. **background-size 是如何工作的**

在我们动手修改页面之前，先看几个简单例子，以便于更好地了解 background-size 的工作机制。

图 3-2 展示了一个宽 200 像素高 120 像素的图片，图 3-3 展示了当在一个宽 500 像素高 200 像素的 div 元素里设置这张图片为正常的重复背景时的表现；由于 div 元素的尺寸和图片的尺寸不是正好是倍数，所以在右侧和底部，图片的一些区域会被裁剪而不能显示。



图 3-2　200 × 120 像素的图片



图 3-3　当图片是重复背景的时候，右侧和底部的部分会被截掉

3

我们可以使用 background-size 属性缩放图片，首先将宽度从 200 变成 100。

```
div {
    width: 500px;
    height: 200px;
    border: 1px solid #999;
    background-image: url(images/stars.gif);
    background-size: 100px auto;
}
```

backgroun-size 属性的第一个值 100px，指定了背景图片的宽度。第二个值 auto，指定了高度。auto 这个值设定了之后能够让高度自动地按照比例缩放。如果你将这个值置空，浏览器会默认使用 auto。因此如果这样设定 background-size: 100px;也能达到同样的效果。对比图 3-4 和图 3-3 就可以看到背景图片已经被缩小了，但是保持了正确的长宽比例。



图 3-4　浏览器将图片缩放成 100px 宽，这样它在宽度上正好适应 div，所以右侧没有被截取

如果你在 background-size 的值里面使用百分比，请注意这个百分比是相对于 div 的宽度，而不是图片本身的宽度，比如我想在 div 里面显示的背景图片重复一次，但是不会产生任何截断，我会使用如下的 CSS：

```
div {
    width: 500px;
    height: 200px;
    border: 1px solid #999;
    background-image: url(images/stars.gif);
    background-size: 50% 100%;
}
```

此时背景图片会被拉伸，宽度上会适配 div 宽度的一半，高度则会和 div 的高度一致，然后再重复一次（图 3-5）。浏览器改变了背景图片的形状，也对它进行了放大，这会让图片的边缘看起来有些模糊和颗粒状。因为是由浏览器来实现缩放，background-size 并不适合所有的图片，只有当图片精度本身不高、比较抽象或者非常简单的时候才能完美适配，这种图片的边缘往往不太清晰，就好像我们的线格纸质笔记背景图片一样。



图 3-5　浏览器将图片缩放到正好重复一次可以填满 div，图片变形了但是没有被截断

更多平铺背景图片的方式

除了使用 background-size 让背景图片正好适配盒容器以外，还可以用 background-repeat 属性里面的 round 和 space 值来使背景图片平铺之后不会在边缘上被截断。这些值是 CSS3 新引入的，可以和 background-size 联合使用，也可以独立使用。

round 值会缩放背景图片，从而保证图片填满整个区域而不会被截断；space 值不会缩放图片，但是它可以保证填满并且第一张和最后一张背景图是紧靠容器边缘的。

不幸的是，在本书写作之时，唯一能够支持这些值的浏览器是 IE 9 和 Opera，而且 Opera 的实现还不尽完整并且问题多多。在 backgroud-repeat 的这些值得到很好的支持之前，我们还是乖乖地用 background-size 吧，虽然它没有 round 和 space 听起来那么好。

2. 让纸质背景图片的线格和文本一起缩放

为了让背景图片和文本一起缩放，我们需要将它的尺寸设置成既不是百分比也不是像素值，而是 em，它是一个以当前的字体高度为衡量基准的相对单位。

开始之前，先从 www.stunningcss3.com 下载练习文件，然后在你的编辑器里打开 paper_start.

html。它的 CSS 定义放在页面头部的 `style` 元素里。

找到 CSS 里面对`#page` 的定义，然后添加 `background-size` 属性，把 Mozilla 和 Webkit 的私有定义也加上去：

```
#paper {
    float: left;
    margin: 40px;
    padding: 3.2em 1.6em 1.6em 1.6em;
    background: url(images/paperlines.gif) #FBFBF9;
    -moz-background-size: auto 1.6em;
    -webkit-background-size: auto 1.6em;
    background-size: auto 1.6em;
}
```

Opera、Chrome、Safari 5、Firefox 4 和 IE 9 会使用标准的 `background-size` 属性；Firefox 3.6 和 Safari 4 及之前的版本会使用`-moz-`和`-webkit-`版本的属性。在属性里面，我们告诉浏览器，希望图片的高度是 `1.6em`，然后宽度应该是自适应的。这意味着从第一条线开始，这张图片上的所有线格之间的距离都是 `1.6em`。为什么是 `1.6em` 呢？因为每行文本的行高是 `1.6`，这是在 body 元素的 `line-height` 属性里面已经定义好了的：


```
body {
    margin: 0;
    padding: 40px;
    background: #CCC url(images/background.gif);
    color: #333;
    font-size: 87.5%;
    font-family: Georgia, "Times New Roman", Times, serif;
    line-height: 1.6;
}
```

图 3-6 显示了背景图片的大小已经发生了改变，但是文本还是没有和背景图片的线格对齐。这是因为还没有将所有文本的大小和边距都设置成可以适配 `1.6em` 的行高。段落和列表的文本已经和背景图片很适配了，因为如你在 CSS 里面看到的那样，它们的行高已经是 `1.6`，底边距也是 `1.6em`。但是标题的值还得调整一下。

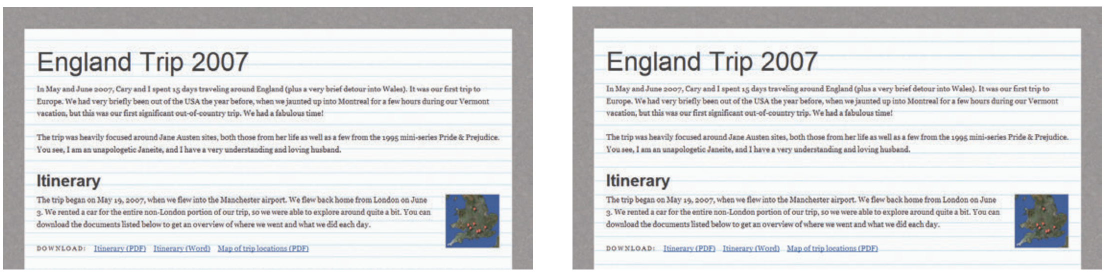


图 3-6 设置了 `background-size` 之后的效果对比

```
h1 {
    margin: -.3em 0 .14em 0;
    color: #414141;
```

```
        font-family: Helvetica, "Helvetica Neue", Arial,
        ¬ sans-serif;
        font-size: 3.5em;
        font-weight: normal;
    }
    h2 {
        clear: left;
        color: #414141;
        margin: 0 0 -.14em 0;
        font-family: Helvetica, "Helvetica Neue", Arial,
        ¬ sans-serif;
        font-size: 2.17em;
        font-weight: bold;
    }
```

这些边距的值是经过多次尝试才得出来的。不像像素值度量那么直接，你没法找到一个能够在所有浏览器上都能显示得非常好的值，每个浏览器将相对值比如 `em` 转换成屏幕上显示的像素值的计算方法都有差异。这里指定的值可以在 Firefox、Safari 和 Chrome 上显示正常。所有的文本都正常地置于 `1.6em` 的线格之内（图 3-7）。



图 3-7　Firefox 3.6 下，文本和线格现在已经很好的对应了

但是在 Opera 上文本还是没有对齐，因为 Opera 对背景图片的调节度要比其他浏览器小。如果我们要调节好 Opera 上的表现，那么又会导致其他浏览器不能正常显示。这里你必须决定哪些浏览器对你更重要，这通常基于网站来访者的统计。

当文本和背景图片适配了之后，如果用户使用了特殊的默认字体大小或者自己对文本的大小进行缩放的时候，背景图片也会相应地匹配文本，这能够让文本和线格始终保持一致（图 3-8）。同样地，当你改变了 `body` 元素里面的基本字体大小时，它们同样也能缩放以保持适配，这样你就不用再去对背景图片进行调节了。



图 3-8　即使用户使用较大的字体，文本和线格还是能保持一致

3. 针对 IE 的兼容方案

`background-size` 属性在 IE 8 及之前的版本上是无法工作的，而且也很难直接模拟实现这个特性。但在这个例子里，它只是一个小的视觉特效，因此从渐进增强的角度来说，我们对它在 IE 下的缺失完全没有担心的必要。

你也可以通过 Modernizr 为 IE 提供一个备选的样式，Modernizr 可以检测到浏览器是否支持

background-size。你完全可以为不支持的浏览器提供一个不同的背景图片，或者可以提供一个设计为可以适配特定像素值字体的线格纸质背景图片，然后对不支持 background-size 的浏览器指定字体大小为特定像素值。我并不建议这么做，因为从可访问性的角度来说，基于像素的字体大小设置是非常坏的做法。然而，当你试图使用 background-size 特性来缩放背景图片时，Modernizr 确实是提供备选样式的不错解决方案。

background-size 属性的来龙去脉

background-size 属性是背景和边框模块的一部分，参见 www.w3.org/RT/css3-background。它有宽度和高度两个值，可以设置成 auto，也可以是任何单位的具体值。

background-size 可以设置成容纳或者覆盖，这两者都可以让浏览器按比例缩放背景图片。当设置成容纳模式时，浏览器会让背景图片尽可能地放大， 但是会确保宽和高都不会超出容器，这能够保证图片不会被截断，但是可能会导致容器的部分背景区域没有图片。当设置成覆盖模式时，浏览器会保证背景图片的宽和高都至少不小于容器对应的宽和高，这通常会导致图片的截断，但是能保证容器的背景区域被完整覆盖。

除了我们上面提到的例子，你还可以在下面一些场合使用 background-size。

- 在流动或者弹性布局中，对头部区域的非重复性背景图片使用这一特性，确保它能始终适配宽度。
- 对于平铺的重复性背景图片，使用这一特性来确保背景图片不会产生截断的效果。
- 制作一个始终填满整个页面的背景图片，参见 www.alistapart.com/articles/supersize-that-background-please。
- 在流动布局中缩放背景图片来伪造出分栏的效果，参见 www.css3.info/liquid-faux-columns-with-background-size。
- 使一个链接或者列表元素的背景图标能够和文本一起缩放。
- 针对 iPhone 4 的高分辨率显示屏，将背景图片进行缩小一半的显示预处理，这样当这块优秀的显示屏以双倍像素值显示时，背景图片不会变得模糊，参见 http://dryan.com/aritcles/posts/2010/6/25/hi-res-mobile-css-iphone-4。
- 使用媒体查询技术来侦测用户的窗口大小，然后据此来缩放背景图片以适配，媒体查询技术将会在第 6 章介绍。

表3-1 **background-size**的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，4+；3.6需要-moz-前缀	支持	支持，5+；3+需要-webkit-前缀	支持

3.2.2 单一容器上的复合背景图片

CSS3 所带来的让 Web 设计师最开心的变化之一就是能够在单一容器上使用复合的背景图

片。在我们这个例子里，会使用这一技术让页面看起来更真实——会添加一些小细节，比如在页面顶部增加一个小图钉什么的。

在 CSS3 之前，每个容器只能指定一张背景图片，因此每当你需要增加一张背景图片的时候，必须至少添加一个容器来容纳它。当然如果你很幸运，HTML 结构里面恰好就有些可用的容器，比如 h3 之类的元素通常会作为嵌套元素使用，你就没必要再添加额外多余的容器了。但是，这样做是有巨大风险的，这意味着页面的表现和内容被绑死了，如果内容发生改变，这些背景图片该放到哪里去呢？

用嵌套 div 容器显示特定背景的做法不是很复杂，但是它明显是难以管理的。它让你的 HTML 标记更加复杂同时也会增加页面文件的大小。如果要增加某个图片效果，你不仅需要修改 CSS，还要修改 HTML 代码。

通过使用 CSS3，HTML 代码可以保持不变，你只需要在 CSS 的 background-image 或者 background 属性中列出需要使用的所有背景图片，然后用逗号分开它们。每张图片都可以定位，设置重复值，改变大小以及其他可以单独控制的特性。

图 3-9 展示了我们想增加到页面上的背景图片。为了把它们添加上去，我在#paper 规则中添加了一个新的 background 的声明：

```
#paper {
   float: left;
   margin: 40px;
   padding: 3.2em 1.6em 1.6em 1.6em;
   background: url(images/paperlines.gif) #FBFBF9;
               background: url(images/thumbtack.png),
               url(images/stains1.png),
               url(images/stains2.png),
               url(images/stains3.png),
               url(images/stains4.png),
               url(images/paperlines.gif) #FBFBF9;
   -moz-background-size: auto 1.6em;
   -webkit-background-size: auto 1.6em;
   background-size: auto 1.6em;
}
```

说明　background 属性里的换行和缩进是为了方便阅读而设，你也可以将它们写作一行，这并不影响代码功能。



图 3-9　添加这五张背景图片将会增加细节效果

旧的 background 声明将会继续被 IE 和其他不支持复合背景图片特效的浏览器所使用。因为它们无法理解新的 background 声明中所使用的语法。而支持这一特效的浏览器将会使用新的声明来重载旧的声明。

复合背景图片也是有层次地显示的，最先声明的图片将会居于最上层，这就是为什么我要先声明图钉的图片，而把线格纸质图片放到最后的原因。

光声明对图片的使用还远远不够，我们还要分别告诉浏览器，对这些图片，应该如何定位，如何重复，以及各自的大小如何。要这么做，只需要简单地把按逗号分隔的定义当做单独的 background 定义来指定就好。图 3-10 展示了可以使用在 background 属性中的各种值，并且这些值的先后顺序是有意义的，所以我建议每次书写 background 属性定义的时候都按照这个顺序，以免混淆或者出错。（至少我就常常弄错！）

说明　图 3-9 所示的水渍图是用 Obsidian Dawn 所开发的 Photoshop 的笔刷绘制而成（www. obsidiandawn. com/water-stains-photoshop-gimp-brushes）。



图 3-10　background 属性可以包含复合层次，顶层的定义包含所有可能的值（除了 color，它只能在底层定义）（另见彩插图 3-10）

接下来我们按照图 3-10 所示，给我们的例子增加背景图片定位、重复方式等：

```
background: url(images/thumbtack.png) 50% 5px no-repeat,
             url(images/stains1.png) 90% -20px no-repeat,
             url(images/stains2.png) 30% 8% no-repeat,
             url(images/stains3.png) 20% 50% no-repeat,
             url(images/stains4.png) 40% 60% no-repeat,
             url(images/paperlines.gif) #FBFBF9;
```

接下来，修改 background-size 属性来告诉浏览器，除了最底层的线格纸质图片以外，其他图片都使用它们原生的尺寸大小。

```
-moz-background-size: auto, auto, auto, auto, auto,
                      auto 1.6em;
webkit-background-size: auto, auto, auto, auto, auto,
                        auto 1.6em;
background-size: auto, auto, auto, auto, auto, auto 1.6em;
```

在这里，每个逗号分隔的值是和 background 属性定义里面逗号分隔的值一一对应的。

虽然从理论上来说，你可以在 background 属性里面直接指定 background-size 的定义，但是这现在还无法正常运行。虽然 Opera、Chrome、Safari 5、Firefox 4 和 IE 9 都可以直接接受 background-size 的定义，但旧版的 Firefox 和 Safari 还是需要加上浏览器前缀的声明，否则它

们就拒绝工作。因此，为了让 background-size 能运行得更好，也为了避免你自己把它和 background-position 搞混（这可是非常容易犯的错误），还是分开定义 background 和 background-size 比较好。

保存好你的代码，然后在够新的浏览器里面测试页面。你将会看到文本和背景线格还是保持一致，但页面上多了污渍的效果，页面的顶部也有了图钉的特效（见图 3-11）。

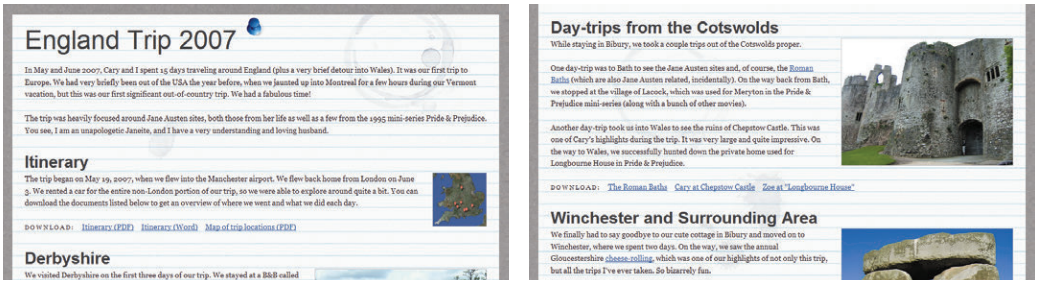


图 3-11　div 容器里面现在能够显示全部 6 种背景图片的效果了

说明　至此为止的所有修改都在 paper_1.html 页面里，它在你所下载的本章的示例文件之中。

通过在一个单一的 div 容器上单独设置每张背景图片的定义，而不是把这些图片直接组合成一张背景图片，你可以让这些图片在页面大小发生变化时有更灵活的布局，不管页面的大小尺寸是怎样的，这些污渍的图片效果是均匀分布的，而不是挤在某个点上。

复合背景图片的来龙去脉

复合背景图片并不是一个单独的属性，它是 background 和 background-image 属性的新特性。这些属性是背景和边框模块的一部分，参见 www.w3.org/TR/css3-background。

在 background 和 background-image 属性的定义中列出每张背景图片的定义，然后以逗号分隔，这些图片是分层展示的，最先列出的图片在最上层。

每张图片都可以单独定位、设置重复模式以及大小。你可以直接在 background 属性里每个图片的 URL 定义的后面直接加入这些定义，或者单独用一个逗号分隔的列表来定义 background 的各个属性，比如这个例子，background-repeat: no-repeat, norepeat, repeat-x, repeat。这个逗号分隔的列表里面的值和 background 里面定义的图片列表保持一一对应就好。

除了我们的例子，你可能在下面一些场合也会考虑使用复合背景图片。

- 弹性容器中需要一些比较花俏或者不规则的边角图片，而其他的 CSS3 特性比如 border-radius 又搞不定的时候，比如一个华丽的按钮；参见 http://css-tricks.com/css3-multiple-backgrounds-obsoletes-doors。

- 当使用图片标记对 `blockquote` 标签的开始和闭合进行修饰的时候。参见 http://css.dzone.com/news/multiple-backgrounds-oh-what-beautiful-thing。
- 通常我们所说的“视差效果”，就是当改变一个窗口的大小或者在某个 `div` 上悬停的时候,希望产生一些图片能够以不同的速度相对移动的效果;参见 www.paulrhayes.com/2009-04/auto-scrolling-parallax-effect-without-javascript。
- 当某些单一图片需要更自然地拉伸以适配整个容器或者页面的时候，你可以考虑把单一图片先打散成为多张图片，然后采用复合背景图片技术，比如以一张风景图片作为背景，但是风景中的太阳始终显示在最右上角，而有棵树始终保持在左下角。
- 当希望把图片均匀的分布在容器里面的时候，可以使用按百分比定位的方式保证这些图片能够按一定的间隙分布，比如把蓝色背景和云朵效果的背景图片复合在一起。
- 网页设计中最常使用的一种效果，比如顶部和底部的图片固定，但中间部分需要有不断重复的效果，现在我们可以把它们全部放在一个容器了。
- 对背景图片上应用 CSS3 渐变效果（别忘了，它使用的是 `background-image` 属性而不是 `background-color` 属性来定义的），可以渲染出特定的材质，但是又可以让边缘保持单一颜色；或者也可以在用户卷动页面的时候显示背景图片的一部分；参见 http://atomicrobotdesign.com/blog/htmlcss/make-the-thinkgeek-background-effect-using-css3。

表3-2　复合背景图片的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，3.6+	支持，10.5+	支持	支持

针对 IE 的兼容方案

IE 8 及之前的版本，还包括旧版本的 Firefox 和 Opera 都不支持复合背景图片。在我们这个例子里，这些图片仅仅起了锦上添花的效果，你可以完全不用考虑做什么兼容工作。使用这些旧式浏览器的用户还是能看到线格纸质背景图片，并不会让他们感觉缺少什么。

但是总有些场合，当缺少一些图片的时候，总体效果会大打折扣或者残缺不全。比如当你使用复合背景图片创作一个按钮的时候，设计好按钮由左中右三个图片切片复合构成，但是在这些旧式浏览器里面却仅仅显示了中间的部分，这个按钮会看起来相当悲催。请在这样的场合谨慎地使用复合背景图片技术，因为你可选的兼容方案实在不多。

- **使用一张可作为备选的单一图片**。对于这些不支持的浏览器，最简单地方案就是提供一张单一的图片。你可以使用 Modernizr 来分别定义不同的浏览器，也可以简单地再重复定义一次 `background-image` 声明，然后把这个声明置于复合背景图片的声明前面（我们已经在例子里用了这个方法了）。你得确保这张备选图片确实能正常工作。这个方案确实挺容易实现，也不会对支持复合背景图片的浏览器造成什么问题。但是相比起复合背景图片来说，它显然让页面的表现力大大下降了。
- **使用嵌套 `div` 来显示复合背景图片**。相比起上一种方案来讲，这种方案更稳健，但是工

作强度也非常大。我们又回到老路上去了，使用嵌套的 `div` 来分别显示不同的背景图片。如果你想这样做的话，必须使用 Modernizr 或者 IE 条件注释来检测浏览器并相应地做不同的处理。否则，在支持复合背景图片的浏览器里面背景就会重复显示。但此时此刻，倒不如完全不使用复合背景图片技术，因为嵌套的 `div` 在所有的浏览器上都能正常工作。因此我甚至都不认为这算得上是一个兼容方案。

- **生成附加元素来显示附加的图片**。实现“嵌套 `div`”的更干净的方法是使用 `:before` 和 `:after` 等伪类来生成附加元素，然后在其中显示背景图片。Nicolas Gallagher 的文章“CSS2.1 下的复合背景和边框”阐述了这种方案（http://nicolasgallagher.com/multiple-backgrounds-and-borderswith-css2）。这种方案在 IE 8 和 Firefox 3.5 上能很好地工作，但是 IE 6 和 IE 7 并不支持这些伪类，这又带来了新的兼容需求——除非使用脚本让这些旧式的浏览器也能支持这些伪类。你还得保证支持复合背景图片的浏览器能够忽略这些伪类的定义。这样我们就把兼容方案搞得复杂无比了，你又一次需要从自己和用户的角度出发来衡量这么做是否值得。
- **使用 `canvas`**。如果你对脚本技术很在行，那你可以考虑使用 HTML5 `canvas` 标签，在单一容器上绘出复合背景。当然 IE 8 及之前版本的浏览器不支持 `canvas`，但用 Google 的“explorer-canvas”的脚本就可支持了（http://code.google.com/p/explorercanvas）。解释如何使用这个脚本已经超出本书的范围了，Hans Pinckaers 的 mb.js 可以帮到你，它使复合背景在 IE 及更旧的浏览器上正常工作（http://github.com/HansPinckaers/mb.js）。

3.2.3　添加图片边框

另外一个很棒的视觉细节是在页面的左边增加一个边框，让它看起来像是从一本笔记本里面撕下来的一页（图 3-12）。在 CSS3 中，有多种方法可以实现这一效果。



图 3-12　撕裂的笔记本效果图片

1. 使用背景图片

我们可以把这张图片直接作为另外一张背景图片添加，只需要设置它在垂直方向重复就好。但是这张图片有些部分是透明的（你可以看到的那些类似装订的洞），因此它下方的图片会有部分通过这些透明的区域透显过来。如果我们的页面的是单一背景色而不是图案，就可以在透明区域填充这种单一背景色，让整个页面的效果看起来比较无缝。但是显然目前我们的页面无法这样做。

当页面不是单一背景色的时候，你唯一的选择就是在纸面的 div 上包裹一个 div，然后在包裹 div 上设置边框的背景图片。你必须给这个 div 足够的 left-padding 来保证内部的 div 不会叠加到边框图片上去。这可不是理想的解决方案，因为需要增加额外的 HTML 标记，但是在所有的浏览器上都能够运行得非常好。

把边框图片设置为背景的一个小缺点是我们没法控制在容器底部这个图片是如何被截断的。它可能正好在洞的中间的时候就截断了，而这看起来不太像一页真正的纸质笔记（图 3-13）。我承认这并没有多严重——只是个非常小的问题而已。但是如果用 CSS 能很容易解决的话，为什么不呢?

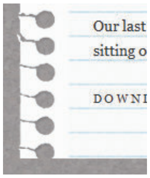


图 3-13　边框图片被截断的效果

CSS3 的解决方案是把边框图片的 background-repeat 值设置成为 round（CSS3 新引入的值）。这会让浏览器尽可能地重复图片以适配容器的尺寸，当如果没法完全适配的时候，浏览器会对图片进行一定的缩放以避免产生底部截断这样的状况。

不幸的是，本书写作之时，只有 IE 9 和 Opera 才能支持 round 值，Opera 的实现还满是缺陷。因此，background-repeat: round 此时还并不是一个真正可用的解决方案。幸运的是，我们可以用 border-image 特性来取代 background-image。

2. 使用 border-image

CSS3 能让我们指定一张图片作为边框，就像我们之前指定一个颜色值或者一种线形一样。浏览器会识别指定的这张图片，然后把它切片，在每条边框上用拉伸或者平铺的方式来显示这些切片。

举例来说，假设我们想将图 3-14 所示的图片应用为一个 div 的边框，使用这张图片的顶部的 30 像素的部分来作为顶部的边框，右边 25 像素的部分作为右边的边框，底部 27 像素的部分作为底部的边框，左边 34 像素的部分作为左边框（图 3-15）。我们需要同时指定边框切片的宽度以及它们的位置。

```
.clouds {
    width: 400px;
    height: 150px;
    border-width: 30px 25px 27px 34px;
    border-image: url(clouds.png) 30 25 27 34 stretch;
}
```

说明　你可以对边框宽度和图片剪切区域使用不同的值。浏览器会对每个图片剪切区域进行缩放以适应与之对应的边框宽度。



图 3-14　该图片的边框是无规律的，因此可以用 border-image 来拉伸并平铺

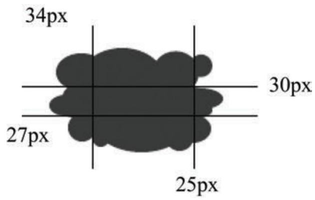


图 3-15　线条表示了我们将如何对该图片进行“虚拟”切割，以便我们在边框中将其拉伸并平铺

border-image 值的第一部分是图片的路径，这和 CSS 中其他的路径值是一样的。

接下来的 1 到多个数字部分就是定义浏览器应该如何对图片进行切片。在我们的例子里，我们使用了 4 个数字，因为希望指定 4 个角上的数值都各有不同。第一个数，30，是对顶部的边框图片切片位移的指定，单位是像素。第二个数，25，是对右边的边框切片位移的指定，第三个数是对底部的指定，第四个数是对左边的指定。浏览器会根据这个定义对图片进行切片，然后产生九张图片，分别对应 4 条边，4 个角和容器的中央部分。

说明　奇怪的是，在 border-image 的切割值中你不能带上 px 单位。或者你可以使用相对于图片本身的百分数来作为切割值，但这一次必须带上百分号。

中央切片

边框图片的中央切片会覆盖整个容器的中间区域，就是整个边框区域所包裹的部分。虽然不太直观，但是确实提供了更多的样式选项。如果你不希望这个中央切片覆盖了你的背景图片或者背景颜色，那么在制作边框图片的时候，请让它的中间部分保持透明，然后将图片保存成为透明的 GIF 或者 PNG。

其实在规范定义中，中央切片默认会被无视，除非你在 border-image 的定义里面加上 fill 关键字。但是，当前没有浏览器支持 fill 关键字，它们默认就给你 fill，却居然不给个 not fill 的选项。

border-image 属性的来龙去脉

border-image 属性是背景和边框模块的一部分，参见 www.w3.org/TR/css3-background。它是一个速写属性，但是现在你还不可以使用各个独立的属性，因为还没有任何浏览器支持 border-image 之外的定义。

在 border-image 的值里面，你需要指定一张图片，以及对图片进行切片的方式和各个切片（除了 4 个角）的重复的方式。

如果你不需要分别对 4 个边框都做特别的设定的话，可以用 1 个数值应用到所有 4 个边框。如果只指定 1 个值，那么这个值将会应用到 4 个边框上；如果指定了 2 个值，那么第 1 个值会应用到底部和顶部，第 2 个值会应用到左边和右边；如果指定了 3 个值，那么第 1 个会应用到顶部， 第 2 个会应用到左边和右边，第 3 个会应用到底部；如果指定了 4 个值，那就是从顶部开始，按顺时针方向分别应用到 4 个边框。图 3-15 展示了浏览器切片的方式。

repeat 值可以设置成 stretch、repeat、round 或 space。如果只使用 1 个 repeat 值，那么会应用到所有 4 个边框；如果指定了 2 个值，那么第 1 个值会应用到底部和顶部，第 2 个会应用到左边和右边。这个值如果指定为 repeat，那么它会导致所有 4 条边，包括中央的切片都会平铺；如果是 stretch，则会把切片拉伸以填满所有区域；round 会平铺缩放好的切片来保证切片正好铺满整个区域；space 则会让切片在整个区域里面均匀地分布。

请记住，border-width 的值总是要伴随 border-image 一起设置，这样才能留出足够的边框空间来显示图片。其实规范里面还有个 border-image-width 的属性，但是目前还没有任何浏览器可以支持，同样也没有浏览器支持 border-image-outset。

不幸的是，border-image 与通过 border-radius 产生的曲线边框兼容性不好。

除了我们的例子，你可能还会把 border-image 应用于以下一些场合。

- 按钮：参见 http://ejohn.org/blog/border-image-in-firefox。
- 渐变背景。
- 锯齿状的边缘，比如邮票或者彩票。
- 用图片边框效果创建一个相框或者证书框之类；参见 www.norabrowndesign.com/css-experiments/border-image-frame.html。
- 曲线或者有角度的边框。
- 曲线或者有角度的阴影（box-shadow 只能实现一般的阴影效果，但是你可以创建一张不规则的阴影图片然后应用到 border-image 上）。

表3-3　border-image属性的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	部分支持，需要-moz-前缀，3.5+	部分支持，10.5+	部分支持，需要-webkit-前缀	部分支持

repeat 值对于浏览器具体是如何实现边框图片的效果起到了至关重要的作用。在这个例子里，我们使用了 stretch 值，这将会导致浏览器把 4 个边框上的切片都拉伸来填满整个区域（图 3-16），包括中央切片（但不包括 4 个角）。你也可以将它设置成 repeat（图 3-17），或者 round（图 3-18），或者 space。（round 值现在只有 Firefox 和 Opera 支持。）目前还没有浏览器支持 space，因此我连个效果图都没法截取。



图 3-16 stretch 效果



图 3-17 repeat 效果



图 3-18 round 效果

3. 应用撕纸边缘图片

我们开始在页面中使用 border-image 来应用撕纸边缘图片作为边框，以便在文章内容 div 上产生如图 3-12 所示的效果。因为仅仅想把图片效果应用到左边框，所以我们指定左边框为 50 像素宽——那正好是图片的宽度——然后把其他边框的宽度设置成 0：

```
#paper {
    float: left;
    margin: 40px;
    padding: 3.2em 1.6em 1.6em 1.6em;
    border-width: 0 0 0 50px;
    background: url(images/paperlines.gif) #FBFBF9;
    background: url(images/thumbtack.png) 50% 5px no-repeat,
                url(images/stains1.png) 90% -20px no-repeat,
                url(images/stains2.png) 30% 8% no-repeat,
                url(images/stains3.png) 20% 50% no-repeat,
                url(images/stains4.png) 40% 60% no-repeat,
                url(images/paperlines.gif) #FBFBF9;
    -moz-background-size: auto, auto, auto, auto, auto,
```

```
                           auto 1.6em;
    -webkit-background-size: auto, auto, auto, auto, auto,
                              auto 1.6em;
    background-size: auto, auto, auto, auto, auto,
                     auto 1.6em;
}
```

边框图片的实践

border-image 属性其实是有点令人混淆的——我不否认这一点。如果在整个的例子练习完成之后你还是有些不确定的话，我强烈的推荐下面这些跟边框图片相关的 Web 工具。

- Kevin Decker 的“border-image-generator”(http://border-image.com/)能让你上传任意图片然后展示它应用为边框图片的效果。你可以改变切片的位移，边框的宽度以及切片重复的方式，然后马上观察这些改变所产生的效果。
- Nora Brown 的“Grokking CSS3 border-image”（www.norabrowndesign.com/css-experiments/border-image-anim.html）使用 5 个预先设置好的图片，然后提供你一些预设的 border-image 值来观察相应的效果。

改变这些参数值，然后即时的观看所产生的视觉效果的改变，这无疑是详细了解这段 CSS 规则工作原理的最佳方式。

接下来，我们将针对 Chrome 和 Opera 使用标准的 border-image 属性来应用边框图片，同时对 Firefox 和 Safari 使用带有浏览器前缀的属性：

```
#paper {
    float: left;
    margin: 40px;
    padding: 3.2em 1.6em 1.6em 1.6em;
    border-width: 0 0 0 50px;
    -moz-border-image: url(images/edge.png) 0 0 0 50 round;
    -webkit-border-image: url(images/edge.png) 0 0 0 50
    ¬ round;
    border-image: url(images/edge.png) 0 0 0 50 round;
    background: url(images/paperlines.gif) #FBFBF9;
    background: url(images/thumbtack.png) 50% 5px no-repeat,
                url(images/stains1.png) 90% -20px no-repeat,
                url(images/stains2.png) 30% 8% no-repeat,
                url(images/stains3.png) 20% 50% no-repeat,
                url(images/stains4.png) 40% 60% no-repeat,
                url(images/paperlines.gif) #FBFBF9;
    -moz-background-size: auto, auto, auto, auto, auto,
                          auto 1.6em;
    -webkit-background-size: auto, auto, auto, auto, auto,
                             auto 1.6em;
    background-size: auto, auto, auto, auto, auto,
                     auto 1.6em;
}
```

除了左边的切片，其他切片的值都设置成 0，我们不需要从顶部、底部、或者右边做任何切

片，只需要从左边切片 50 像素，所以实际上整个 50 像素宽的图片都用于左边框。

对于 repeat 值，我们使用 `round` 来保证图片能够重复但是也会缩放以保证某个洞不会被截断一半。Safari 和 Chrome 不能支持 round 值，它们会使用尚可接受的 `repeat` 值来作为备选。

4. 使用 `background-clip` 来定位图片

我们的边缘图片现在在 `div` 的左侧已经重复显示出来了，但是背景图片却显示在它的下层（图 3-19）。这是因为，边框是默认绘制在背景区域的上层的。你可能从来没有注意到这个细节，因为通常你的边框都是实心线，不会有透明的部分，但是如果你把 `border-style` 设置成 `dashed`，就会明白我的意思。边框图片同样也是绘制在背景区域的上层的。

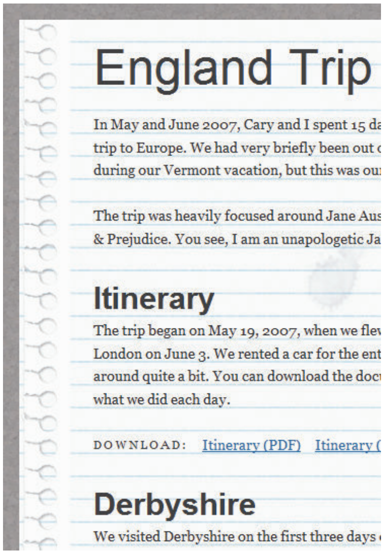


图 3-19　边框图片叠加在背景图片之上

`background` 属性的顺序

通常来说，在每条规则里面书写属性的顺序是没有相关性的，那只是我们经常使用的标准顺序而已，你可以按照自己的意愿来随意安排这些顺序。但是，对于 `background-clip` 属性来说，则必须确保它在 `background` 这个速写属性的后面，因为 `background-clip` 属性是可以直接作为一个值书写在 `background` 属性里面的（参见图 3-10）。如果你先书写了 `background-clip` 属性之后，然后书写了 `background` 属性，但是没有在其中提供 `background-clip` 的信息，实际上是在告诉浏览器你希望使用 `border-box` 的默认值，这会重载你之前所设定的 `background-clip` 的值。

为何不直接把 `background-clip` 的值直接写在 `background` 里面呢？我们现在还不能这样做，正如不能把 `background-size` 的值写进来一样：有些浏览器需要带上前缀的值，无论它在自身还是属性简写中都还不理解这个标准属性。

background-clip 属性的来龙去脉

background-clip 属性是背景和边框模块的一部分，参见 www.w3.org/TR/css3-background。它控制了具体在容器的哪些部分绘制背景。

目前这个属性可以接受的值有 border-box（这是默认值，在边框之下绘制背景），padding-box（背景延伸到 padding 的外边缘，但是不会超出边框的范围），以及 content-box（背景仅在内容区绘制，不会超出 padding 和边框的范围）。Firefox 3.6 及之前的版本不支持 content-box，并且使用 border 和 padding 作为值，而不是 border-box 和 padding-box；Firefox 4 没有这些问题。Safari 5 能够在标准属性中支持 border-box 和 padding-box，但是只在-webkit-background-clip 属性中支持 content-box。

Webkit 还支持一个特殊值 text，但是在-webkit-前缀的属性里，这个值可以让文本作为背景的遮罩，不在文本背后的背景则不可见。这个特性非常酷，但是可能没机会加入标准。如果想获取更多信息或者更多例子，参见 www.css3.info/webkit-introduces-background-cliptext, http://trentwalton.com/2010/03/24/css3-background-clip-text, 和 http://trentwalton.com/2010/04/06/css3-background-clip-font-face。

除了把背景从边框下移出以外，你可能在以下一些场合也会需要 background-clip。

- 从 dashed 或者 dotted 的边框下移出背景色或者背景图片。
- 创作一种双边框的表现，一个是真正的边框，另外一个实际上是 padding，这可以通过使用 content-box 值来实现。
- 通过使用 padding-box，保证背景色不会溢出到圆角边框之外，这情形常常发生在 Webkit 核心的浏览器上；参见 http://tumble.sneak.co.nz/post/928998513/fixing-the-background-bleed。

表3-4　复合背景图片浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，4+；部分，1+，需要-moz-前缀	支持	支持，3+，需要-webkit-前缀；部分，5+	支持

幸运的是，我们能够使用 CSS3 改变这个默认表现。CSS3 使用 background-clip 属性帮助你控制背景和边框之间的相对位置。默认值是 border-box，这就是我们常见的，背景可以一直延伸到边框之下的情况。如果将 background-clip 设置成为 padding-box，就会让背景延伸到 padding 区为止，不会超出到边框之下：

```
#paper {
    float: left;
    margin: 40px;
    padding: 3.2em 1.6em 1.6em 1.6em;
    border-width: 0 0 0 50px;
    -moz-border-image: url(images/edge.png) 0 0 0 50 round;
    -webkit-border-image: url(images/edge.png) 0 0 0 50
    ¬ round;
```

```
    border-image: url(images/edge.png) 0 0 0 50 round;
    background: url(images/paperlines.gif) #FBFBF9;
    background: url(images/thumbtack.png) 50% 5px no-repeat,
                url(images/stains1.png) 90% -20px no-repeat,
                url(images/stains2.png) 30% 8% no-repeat,
                url(images/stains3.png) 20% 50% no-repeat,
                url(images/stains4.png) 40% 60% no-repeat,
                url(images/paperlines.gif) #FBFBF9;
    -moz-background-size: auto, auto, auto, auto, auto,
                          auto 1.6em;
    -webkit-background-size: auto, auto, auto, auto, auto,
                             auto 1.6em;
    background-size: auto, auto, auto, auto, auto,
                     auto 1.6em;
    -moz-background-clip: padding;
    -webkit-background-clip: padding-box;
    background-clip: padding-box;
}
```

Chrome、Safari 5、Firefox 4 和 Opera 都使用标准属性，但是 Firefox 3.6 及之前的版本以及 Safari 4 及之前的版本会使用前缀版本的属性。需要注意的一点是 `-moz-background-clip` 属性会使用 `padding` 而不是标准的 `padding-box` 的值。

这个变更使得背景图片从边框图片的下方移走了（见图 3-20）。

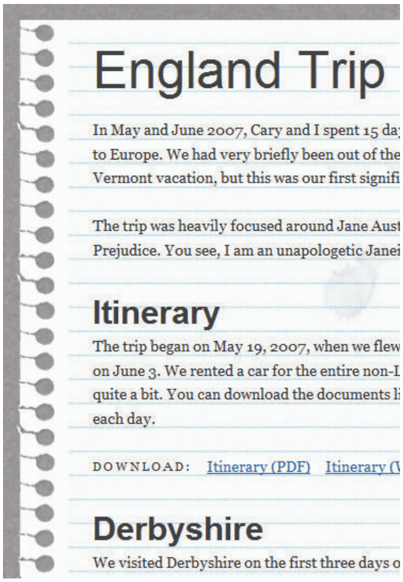


图 3-20 应用了 `background-clip` 之后的效果

5. 针对其他浏览器的兼容方案

不支持 `border-image` 的浏览器根本就不知道究竟发生了什么，它们还是只能显示正常的

线格纸质背景图片。如果你希望加上撕纸边缘效果，像我们之前所讨论的那样，就只能考虑添加一个额外的包裹 div，然后在那之上应用背景图片。

如果你真这么做，有两个选择，一是在所有浏览器上都完全移除 border-image 效果，二是对支持 border-image 的浏览器隐藏兼容的背景图片。我喜欢第二种方式，因为它更灵活，但又不需要特别多的额外工作。只需要简单地运用 Modernizr 或者 IE 条件注释创建一个只有特定浏览器才能看到的包裹 div 就好。如下是这个 div 需要的 CSS 定义：

```
#wrapper {
    padding-left: 50px;
    background: url(images/edge.png) repeat-y;
}
```

其他浏览器不会知道这条规则的存在，它们的 HTML 代码里面自然还是会存在这个多余的包裹 div，但是不会有任何 CSS 应用到它之上。

还有一种可选方案是你可以把线格纸质背景图片和撕纸边缘图片合并成一张背景图片。那样的话就无需增加一个额外的包裹 div，但是会带来更大的工作强度，因为要为不同的浏览器制作不同的图片。你还是得确保支持的浏览器会使用两张图片，一张用于 background，一张用于 border-image。

存在几种方法可以用脚本的方式来模拟 border-image，这样就不用退化到背景图片的方式。但是脚本的方式只支持以 stretch 的方式来显示 border-image，不能是 repeat 也不能是 round，所以脚本的方式并不适用于我们的例子。但如果你的项目只需要 stretch 方式的 border-image 的话，请参见如下解决方案。

- Jason Johnston 的 PIE（http://ccs3pie.com），在第 2 章里有描述。PIE 还包括对 border-image 从 IE 6 到 IE 8 系列的有限支持。
- Louis-Remi Babe 的 borderImage（http://github.com/lrbabe/borderimage）。这是个模拟 border-image 的 jQuery 插件，对于 IE 浏览器，它使用 VML，对于非 IE 浏览器，它使用 canvas。在这里你可以找到更多关于如何使用的详细描述 www.lrbabe.com/sdoms/borderImage。

3.2.4 添加阴影

在第 2 章中，我们介绍了使用 box-shadow 属性来为容器创建阴影效果。纸质笔记页看起来是应用这个效果的绝佳例子，那么我们来尝试添加一下。但是请当心这一点：阴影效果和参差不齐的边框图片无法配合，它只能和整个容器平滑的边适配。这就意味着如果阴影效果应用在我们例子的左边，你只会得到一个直边的阴影（图 3-21），而不会有完美适配撕裂的纸质边缘的阴影效果。

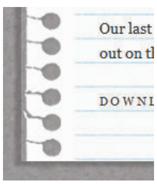


图 3-21 阴影效果遇到毛糙的边缘的时候的效果

为了避免这个问题，在我们的例子里面，需要将阴影设置得离左边足够远。为#paper 增加如下规则：

```
-moz-box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
-webkit-box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
box-shadow: 6px 5px 3px hsla(0,0%,0%,.2);
```

说明 至此为止的所有修改都在 paper_2.html 页面里，它在你所下载的本章的示例文件之中。

这将会在右下的边缘创建阴影效果（见图 3-22）。

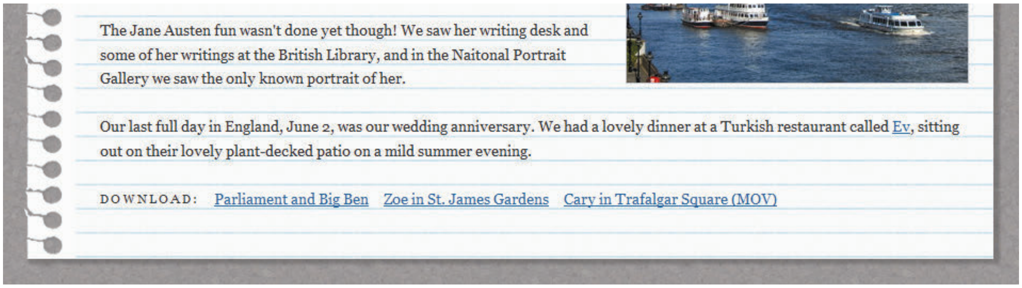


图 3-22 阴影效果

由于 Safari 和 Chrome 并不支持使用 round 作为 border-image 的 repeat 值，那就有可能在这些浏览器上正好有对撕裂效果截断一半的情况，这会让阴影效果看起来有点奇怪（见图 3-23）。这个问题并不大，但是如果它让你很纠结，请移除-webkit-box-shadow 的声明好了。（有时如果每个浏览器都有自己独特的声明的话，世界反而会变得美好，是不是？）



图 3-23 当发生截断时的奇怪效果

当然，现在 Webkit 核心的浏览器上已经没有阴影效果了，为了实现阴影效果而又不使用-webkit-box-shadow 属性的话，你可以创建一张阴影图片，然后使用-webkit-border-image 属性把图片添加到右侧和底部。

3.3 嵌入字体

我们已经在文章的背景上玩了很多花样了。现在让我们对实际的内容添加一些额外的样式。可以使用@font-face 规则来让标题看起来像是手写的，这个戏法在 IE 上也能玩。

3.3.1　@font-face是什么

@font-face 是链接服务器上的字体的一种方式（就像你指定图片链接一样），浏览器会根据这条指令把对应的字体文件下载到本地缓存，使用它去修饰文本。这通常被称为*字体嵌入*（虽然实际上并没有什么字体被嵌入了），这些“嵌入”的字体就是 Web 字体。

这条规则实际上是 1998 年所制定的 CSS 2 的一部分，但是在 CSS 2.1 里面它被移除了，现在在 CSS3 里面它又回来了，并且最终获得了广泛的浏览器支持。

直到现在为止，如果没有 Web 字体，Web 设计师对字体的使用还局限于一个可能在所有的用户机器上都会存在的小的字体集合，它们被称为 *Web 安全字体*。设计师如果不满足于仅仅使用 Arial、Verdana 或者 Georgia（或者其他少数）字体的话，它们将只能依赖图片、Flash 或者脚本技术才能够使用其他特定的字体。所有这些模拟技术都会对网站的可访问性和可用性造成很大的问题。从实现和维护的角度来说，它们也有很大的工作强度，某种程度上也会降低页面的性能。

相反地，如果使用@font-face，将不会对页面上真实的内容文字造成任何影响。你不需要关心用户是否安装了 Flash 插件或者是否开启了 JavaScript。不需要创建任何图片或者脚本，用户也无需下载它们。你的工作简单到只需要编写类似如下的 CSS 规则：

```
@font-face {
    font-family: Raleway;
    src: url(fonts/raleway_thin.otf);
}
h1 {
    font-family: Raleway, "HelveticaNeueLt Std Thin",
    ¬ "Helvetica Neue Light", "HelveticaNeue-Light",
    ¬ "Helvetica Neue", Helvetica, Arial, sans-serif;
}
```

这告诉浏览器使用 raleway_thin.otf 字体文件来渲染 h1 标签内的字体（图 3-24）。如果用户的浏览器不支持@font-face，或者因为某些原因没有办法下载字体文件。浏览器的兼容工作简单到只需要从字体栈查找下一个可选字体来使用。*字体栈*就是在 font-family 属性中声明的字体列表，浏览器会按照声明的顺序尝试从用户的机器上加载这些字体，直到找到一个可用的字体为止。

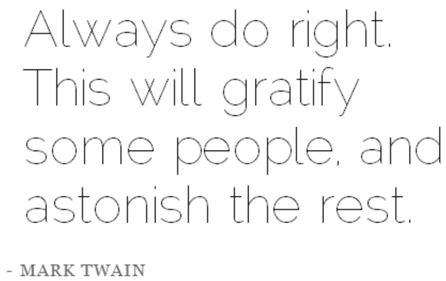


图 3-24　使用@font-face 显示非标准字体

也许你有点将信将疑，正如你所想，在实际工作中使用@font-face 远比上面所说要复杂许多。

说明 欲了解如何制作或找到字体栈和其他资源，请参阅 http://nicewebtype.com/notes/2009/04/23/css-font-stacks。

3.3.2 选择可用字体

最重要的问题之一就是并不是所有的字体都可以在 Web 页面中使用，其中有一些有非常严格的版权限制禁止这样的使用，而另外有一些则在 Web 上看起来不太美观。

1. 版权问题

每次想使用一个字体文件的时候，都要了解一下它的授权文件（通常称为最终用户许可协议（EULA）或者使用条款），看看它是否允许作为 Web 字体被嵌入。许多字体授权都不允许这样做。因为当你使用`@font-face`的时候，字体文件会被下载到用户的本地缓存，就好像图片一样。用户可以到本地缓存目录中去复制字体并自行安装使用。大部分的字体提供者可并不乐意那些访问过你的 Web 站点的人都可以随意地使用字体。

3

说明 有些方法可以使你制作的字体文件更安全。请参阅 http://subjectiveobject.com/2009/10/28/securing-font-face 和 http://typefront.com/。

当然，没几个用户真的会去那么做。但是 Richard Fink 在他的文章“十字路口的 Web 字体”（www.alistapart.com/articles/fonts-at-the-crossing）中描述了字体开发者对 Web 字体嵌入所担心的更大问题：

> 最大的担心来自于一旦字体可以从 Web 上访问，他们也将变成字体销售的一种形式，当前的商业模式会被打破，而且通常来说这也会导致字体的贬值。字体设计者再也没办法像这样和客户开价了：4 种样式的字体，供 6~10 人使用，总价格是 420 块钱。现在字体已经在网站上了，理论上任何用户都可以不花一分钱去使用。如果 Web 仅仅是让最终用户从中收益但是对生产者却没有任何帮助，想想这行得通吗?

更不幸的是，书写大多数字体的授权协议时，大家对`@font-face`毫无概念，所以当你阅读这些授权协议的时候，将看不到任何关于嵌入字体的内容。限制的缺失并不意味着你就可以自由使用这些字体。除非协议明确表明 Web 嵌入或者再发行字体是合法的，你才可以这么做。

说明 http://webfonts.info/wiki/index.php?title=web_fonts_licensing_overview 是一份有关哪些厂商的字体可以被嵌入及其替换方法的总览性文章，尽管如此你在使用前仍旧需要确认每一个字体的协议。

甚至对于免费字体来说，也是这样。因为你能够免费获得字体并使用并不代表你能够将它再次发行。电脑上已经安装的字体亦是如此。

好消息是有很多地方可以找到允许 Web 嵌入的字体。

- The League of Moveable Type（www.theleagueofmoveabletype.com）是一个成长中的小型组织，专门收集可以提供`@font-face`使用的免费或开源的字体。图 3-24 所使用的 Raleway 字体就是来自这个组织的收集。
- Webfonts.info 的维基有一个页面称为“**可供@font-face 嵌入的字体**”（http://webfonts.info/wiki/index.php?title=Fonts_available_for_%40font-face_embedding），上面列出了一些允许嵌入的字体，大部分都是免费的。但是跟很多的维基页面类似，它的更新总是不及时的，所以覆盖范围也不全。
- Font Squirrel（www.fontsquirrel.com）提供了大量的免费字体，这些字体的授权都是允许嵌入的。它还提供了一些小工具来辅助使用`@font-face`，稍后我们会介绍一下。
- Google 有一个允许嵌入的免费字体库，名为 **Google Font Directory**（http://code.google.com/webfonts）。当你通过 Google 字体 API 直接链接这些字体的时候，你使用的是 Google 服务器上存放的字体，这种方式有大量的优点（参见：http://mindgarden.de/benefit-of-thegoogle-font-api）。但是你也可以从 http://code.google.com/ p/googlefontdirectory/source/browse 直接下载字体文件到自己的服务器上。
- Kernest（www.kernest.com）上的字体大部分是免费的，全部字体都允许嵌入。有些字体直接通过 Kernest 网站就可以使用，但大部分都可以下载到自己的服务器上。
- exljbris（www.josbuivenga.demon.nl）和 Fontfabric（http://fontfabric.com）都提供一些可供嵌入 Web 的字体。根据 EULA 授权你可以任意地使用。
- Fonthead（www.fonthead.com）上的所有字体都允许通过`@font-face`进行 Web 嵌入。
- FontSpring（www.fontspring.com/fontface）出售可供传统、印刷用途的字体，也包括供`@font-face`使用的 Web 字体。
- FontShop 创作了一些字体的 Web 版本，称为 **Web FontFonts**（www.fontshop.com/fontlist/n/web_fontfonts），你可以单独购买这个版本而不需要为传统版本付费。

说明　这里所列举的资源很可能发生变化和增长。欲获取最新的 Web 字体资源列表，请访问 www.stunningcss3.com/resources。

避重就轻

我所列出来的所有`@font-face-reedy`字体，你都可以下载并放到自己的服务器上，然后对它们进行编码以提供服务。但是还有一个选项是让别人来干这些重活累活，你只需要使用他们提供的字体嵌入服务，也称为字体分发服务，或者叫字体混淆与伺服服务（FHOS）。

这些服务提供一系列的字体的分发，这些都是授权许可了可供 Web 使用的，这很好地避免了`@font-face`的版权问题。字体都由这些服务的主机伺服，让下载和再发行字体变得困难甚至不可能。

字体嵌入服务非常容易使用，因为他们为不同的浏览器提供了所有必需的文件格式，包括你把字体添加到浏览器所需要的代码。除了 CSS 之外，他们可能还提供 JavaScript 以便让字体不能再重用，或者加快字体渲染的速度。这些服务大都不是免费的，虽然有些服务的有些内容可以免费，他们的定价策略也是很多样化的，例如可以订购一个字体的组合，或者单独为一种字体在一个网站上的应用付费。

这些服务现在非常流行——许多字体生产商仅仅为自己创作的字体提供服务——但是也存在一些大厂商。

- ❑ Typekit（http://typekit.com）是一家订购型的字体服务机构，你可以为你需要的字体集按年付费，这些字体是来自不同厂商的产品。有一个最基本的字体集合是免费的，但是有其他的用户限制。
- ❑ Fontdeck（http://fontdeck.com）是一家订购型的字体服务机构，但付费方式是按照每个网站一年为每个字体单独付费，而不是为一个字体集合付年费。它的字体也来自多家生产商。
- ❑ Kernest（www.kernest.com）的字体订购模式和 Fontdeck 的非常相似，但是它的字体差不多都是免费的。它也拥有多家厂商的产品。Kernest 为一些字体提供了伺服服务，它的大部分字体都可以下载下来自己伺服。
- ❑ Acender 提供两种服务：Web 字体服务（www.ascenderfonts.com/webfonts）以及 FontsLive（www.fontslive.com）。这两种服务都拥有多家厂商的产品，并且订购模式和 Fontdeck 比较类似。
- ❑ WebINK（www.extensis.com/en/WebINK）有和 Typekit 比较类似的订购模式，但它是按月付费的，并且除了字体的本身价格之外，你还得为使用的带宽付费。字体也集合了多家厂商。
- ❑ Webtype（www.webtype.com）拥有和 Fontdeck 相似的订购模式，但是价格主要是基于你所使用的宽带。它同样集合了多家厂商的产品，你还可以另行购买传统版本的字体用来下载并应用到桌面系统。
- ❑ Typotheque（www.typotheque.com/webfonts）只提供它自己制作的字体产品，而且每种字体你只需要付一次费用。
- ❑ Just Another Foundry（http://justanotherfoundry.com/webfonts）也只提供自己制作的字体产品，但是它是按年付费的订购模式。
- ❑ Fonts.com Web Fonts（http://webfonts.fonts.com/）有着和 Typekit 类似的订购模式，但是是按月付费的。最高价格的套餐是允许你下载字体并应用到本地桌面的。但是只有在他们服务的网站上。

如果你想要考虑使用以上任何一家的服务，请先读一下这篇文章“十字路口的 Web 字体”（www.alistapart.com/articles/fonts-at-the-crossing）最后给出的购买建议列表。如果你想获得最新的字体嵌入服务商信息，请来这里 www.stunningcss3.com/resources。

2. 可读性和渲染问题

当你扫除了版权障碍之后，先不要太过于兴奋，现在还不是马上在页面中使用各种奇形怪状的字体的时候。每次当你使用一种 Web 字体的时候，都要有一个明确的需要使用它的原因，而不是仅仅因为它看起来很酷。一定要确保字体的使用确实增强了文本内容的可读性。

你需要使用实际的文本内容来测试 Web 字体是否工作正常。图 3-24 中所使用的 Raleway 字体对于大标题来说非常适用，但是对于正文来说，它太细了，当使用在正文中时，没法很好地渲染，可读性也会降低。大部分商业字体的设计导致了它们在屏幕上没法以较小的形式呈现。因此在很多情形下，最好只将`@font-face` 所引入的字体用于标题，对于正文还是继续使用 Web 安全字体比如 Georgia 或者 Lucida 等。

提示 名为 Soma FontFriend 的小书签（http://somadesign.ca/projects/fontfriend）可以让你便捷地测试字体栈里的字体，它同样支持 Web 字体，这样你就能直观地了解每个字体在你的页面表现如何了。

使用 Web 字体所需要注意的另外一个方面是这些字体的抗锯齿和精调方式会影响到它们的可辨认性。Web 字体在使用了抗锯齿的情况下往往比传统字体的边缘更粗糙，通常这是因为它们并不是为在屏幕上显示而设计的。高质量的字体，跟专门为 Web 设计的字体一样，会有更好的精调设置，简单来说，就是字体文件中的一组定义细则，旨在优化字体边缘使得它们在电脑屏幕的像素网格上显示时更加平滑，从而对人眼的视觉效果会更好。字体格式的作用也是这样：TrueType 格式的字体通常比 OpenType CFF 字体有更好的精调。锯齿的程度不仅仅只与字体文件本身相关，它和操作系统甚至浏览器都有很大关系：Mac 通常都比 Windows 平滑，但是也会稍微模糊一点。Windows XP 的用户如果没有开启 ClearType 的话，字体效果简直不堪入目。ClearType 是微软公司推出的提高屏幕上文本渲染效果的技术。

除了 Web 字体的可读性非常重要以外，字体栈里面的备选字体的可读性也非常重要。请确保测试当 Web 字体不可用时，用户还是可以看到可读的、有吸引力的字体。通常你应该选择和你的 Web 字体比较类似的 Web 安全字体来作为备选。这样的话，不管最终用户看到的是什么字体，你应用到字体上的大小、粗细以及其他样式设置都能够很好地工作。

说明 你可以使用 `font-size-adjust` 属性来强制不同尺寸的字体统一大小，但目前只有 Firefox 支持该属性。欲获取更多信息，请参阅 http://webdesignernotebook.com/css/the-little-known-font-size-adjust-css3-property 以及文末的链接。

多关于精调和抗锯齿的细节

字体精调和抗锯齿是一个远远超越本书范围的技术问题，如果你想了解更多的话，可以看看下面的一些文章。

- Thomas Giannattasio 的“字体抗锯齿的烦恼”（www.smashingmagazine.com/2009/11/02/the-ails-of-typographic-anti-aliasing）介绍了抗锯齿、精调、子像素渲染的相关内容，以及不同的操作系统和浏览器是如何处理 Web 字体渲染的。
- Richard Fink 的“字体设计专家眼中的字体精调”（http://readableweb.com/font-hinting-explained-by-a-font-designmaster）以及 Peter Bil'ak 的“字体精调”（www.typotheque.com/articles/hinting）对于字体精调的工作方式提供了更多的细节。
- Joel Spolsky 的“字体平滑、反锯齿及子像素渲染”（www.joelonsoftware.com/items/2007/06/12.html）比较了苹果和微软对于平滑屏幕文本的不同方式。[①]
- Thomas Phinney 的“浏览器选项与字体渲染”（www.thomasphinney.com/2009/12/browser-choice-vs-font-rendering）解释了浏览器对文本的渲染和操作系统之间的关系。
- Webkit 内核的浏览器能够让你通过对`-webkit-font-smoothing`属性的设置来控制抗锯齿的模式。Tim Van Damme 的“-webkit-font-smoothing”（http://maxvoltar.com/archive/-webkit-font-smoothing）以及 Dmitry Fadeyev 的“字体平滑”（www.usabilitypost.com/2010/08/26/font-smoothing）都对此有所讨论。

说明　幸运的是，Web 字体渲染技术正在不断向前发展。比如 IE 9 使用微软 DirectWrite API 来渲染文字，使 Web 字体看起来非常圆滑。同样地，越来越多的字体供应商也正在贩售 Web 字体，随着`@font-face`的流行，无疑我们将看到更多面向 Web 的付费字体的出现。

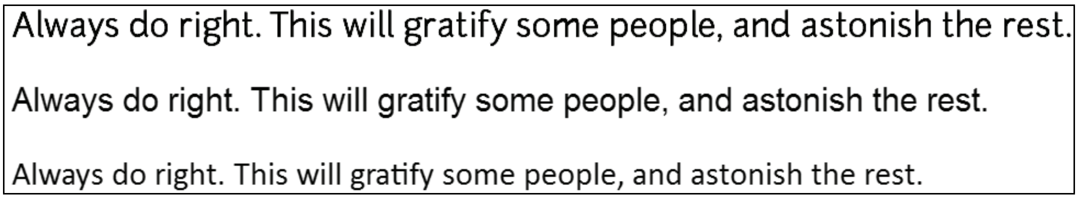


图 3-25　中间的 Arial 和底部的 Calibri 字体都太细小了，不能作为 Web 字体 Junction 的首选替代方案

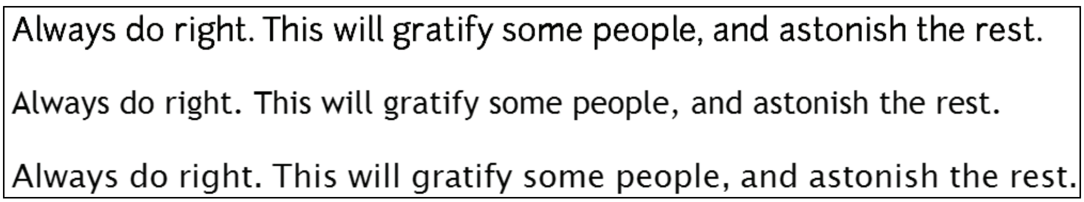


图 3-26　Trebuchet MS 和 Lucida Sans Unicode 都可以作为 Junction 的备选方案

① 该文已收录于人民邮电出版社出版的《软件随想录：程序员部落酋长 Joel 谈软件》一书中。——编者注

3.3.3 浏览器支持度

当你选择了有正确的授权协议的字体，并且能够在屏幕上平滑地显示之后，那接下来的工作就是使用`@font-face`规则来链接这个字体文件并使用，对吗？还没这么简单。`@font-face`在各种浏览器上都得到了很好的支持，但是不同的浏览器会要求你使用不同的字体文件类型。

TrueType（TTF）和 OpenType（OTF）字体文件，跟那些早已经存在于你的电脑上的文件一样，在绝大多数浏览器上都能够正常工作。

IE 浏览器从第 4 版的时候就开始支持`@font-face`了，但是从 IE 4 到 IE 8 都要求你使用特殊的字体文件格式，我们称之为嵌入式 OpenType（EOT）。EOT 实际上不属于一种字体文件格式，它是使用数字版权管理（DRM）协议压缩 TTF 字体之后生成的一个备份，主要用来保护字体不被重用。

唯一能够在 iOS 的 Safari（就是 iPhone、iPod Touch 和 iPad 上所使用的浏览器，也叫做“Mobile Safari”）上正常工作的字体文件格式是 SVG（可缩放矢量图形）。SVG 在 Chrome、桌面 Safari 以及 Opera 上也能正常工作，但在 Firefox 上不行。你可能对于 SVG 作为矢量图片格式的角色非常熟悉，但是 SVG 文件同样能够包含字体信息——实际上，每个字体的元素确实都是由矢量绘制而成的。

说明 欲获取更多关于 SVG 字体的信息，请参阅 Divya Manian 的“About Fonts in SVG”一文（http://nimbupani.com/about-fonts-in-svg.html）。

使用这三种格式——TTF 或者 OTF，EOT 以及 SVG——基本上能够让所有支持`@font-face`的浏览器正常地显示你需要的特定字体，但是为了向后兼容，你也应该了解一下第四种格式，WOFF。

说明 你可以从 www.w3.org/fonts/woff-fAQ 获得更多关于 WOFF 的信息。

WOFF 是在 2009 年引入的，它的含义是 Web 开放字体格式。跟 EOT 类似，WOFF 从技术上来说并不是一种字体格式，它是 TTF 或者 OTF 格式的压缩封装。跟 EOT 不同，WOFF 并不包含 DRM。到现在为止，支持 WOFF 格式的浏览器只有 Firefox 3.6 及其之后的版本、Chrome 6 和 IE 9，但是主流浏览器都在添加对它的支持，很多字体厂商也开始提供这种格式。WOFF 规范在 2010 年 7 月成为了 W3C 的工作草案，因此可以说它得到了官方的支持，有望成为标准的 Web 字体格式。也许在未来，它是唯一的格式。

说明 EOT 和 WOFF 文件使用无损压缩。这些字体与其 TTF 或 OTF 格式的源文件效果一样。

别被这些术语和浏览器搞得太紧张了，在下一节我们就会了解到，创建你所需要的所有格式是非常简单的。表 3-5 是浏览器对字体格式支持的一个总结。

说明　它的浏览器支持度是 CSS 中变化较快的部分。请参阅 http://webfonts.info/wiki/index.php?title=%40font-face_browser_support 和 www.stunningcss3.com/resources 获取最新信息。

表3-5　@font-face文件类型的浏览器支持度

	WOFF	OTF	TTF	SVG	EOT
IE	9	不支持	9	不支持	4
Firefox	3.6	3.5	3.5	不支持	不支持
Opera	不支持	10	10	10	不支持
Opera Mobile	不支持	9.7	9.7	9.7	不支持
Safari	不支持	3.1	3.1	3.1	不支持
Chrome	6	4*	4*	0.3	不支持
Safari on iOS	不支持	不支持	不支持	3.1	不支持

*Chrome 3支持OTF以及TTF字体，但不是默认支持——你必须执行一个命令行命令才可以。表格中列出的浏览器版本是指这个浏览器最早开始支持的版本，所以这个版本之后肯定是支持的。

3.3.4　字体格式转换

有一些字体的提供商会提供给你所有不同浏览器所需要的不同的字体格式。举例来说，Font Squirrel 提供了一个称为“@font-face 字体集”的东西，里面包含了字体的原始 TTF 和 OTF 格式，也有 SVG 版本、WOFF 版本以及 EOT 版本，然后还有一个简单的 HTML 页面以及样式表，示范你应该在 CSS 里面添加哪些@font-face 规则。可以在这里下载这个字体集 www.fontsquirrel.-com/fontface。

更厉害的是 Font Squirrel 的@font-face 字体集生成器（www.fontsquirrel.com/fontface/generator）。你可以上传你的字体，并且生成需要的任何格式。你也能控制它输出的 CSS 语法，通过减少一些内容来减小文件大小，然后使用更多的选项调优字体（见图 3-27）。

Font Squirrel 所生成的文件通常就是你全部所需，但是还有一些值得一提的工具可以优化你的 EOT 和 SVG 字体文件。EOTFAST 是一款免费的桌面软件（在 http://eotfast.com/下载），它可以将 TTF 文件转换成为无损压缩的 EOT 文件；Font Squirrel 所生成的 EOT 文件是未压缩的。命令行工具 ttf2svg（http://xmlgraphics.apache.org/batik/tools/font-converter.html）可以将 TTF 文件转换成 SVG 文件，转换之后文件大小保持不变，甚至有可能更小；你必须安装 Java 以及 Java SVG 工具套件。

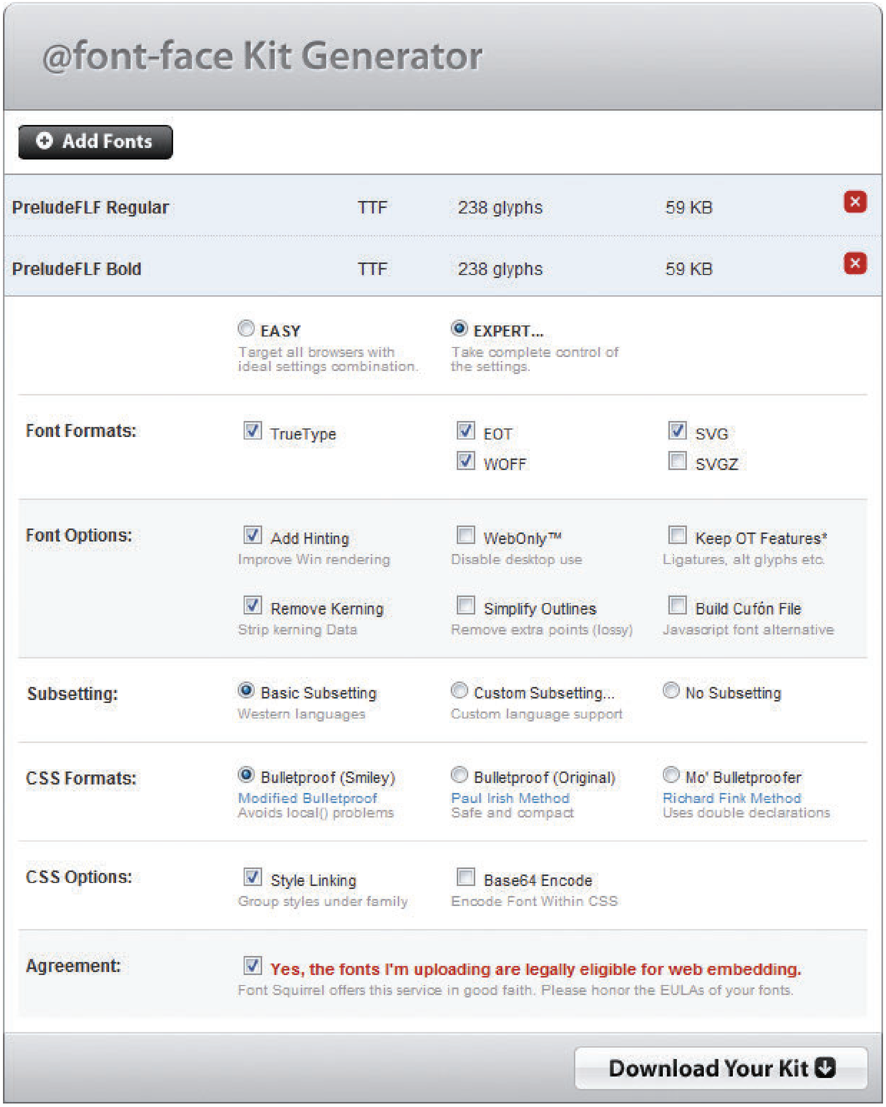


图 3-27 Font Squirrel 的`@font-face`字体集生成器

3.3.5 使用`@font-face`

现在让我们正式使用`@font-face`。既然要打造纸质笔记效果，那么手写字体肯定是必不可少的。我为标题文本选择了 Prelude，一个比较随意的手写字体（见图 3-28）。我们不会对正文也使用这种字体，因为在一个很小的尺寸的时候这种字体会变得很难看而且可读性很差。



图 3-28　Font Squirrel 网站上的 Prelude 字体

在本章的练习文件里面，你会发现一个名为“fonts”的文件夹，里面包含了我们的页面需要的所有 8 个版本的 Prelude 字体文件：EOT、SVG、TTF、WOFF，各自都同时有正常和粗体两种。我使用 Font Squirrel 的生成工具创建了这些版本，使用了图 3-27 中所示的设置。我还使用 EOTFAST 重新制作了 EOT 格式，这将它的大小几乎缩小了一半。

1. 使用`@font-face`规则来链接文件

你可能会注意到图 3-27 中 Font Squirrel 的生成器对于 CSS 格式总共有 3 个选项，分别对应 `@font-face` 的三种 CSS 语法格式。在 CSS 代码中，`@font-face` 的多种语法格式都有着同样的效果，所有这三种语法都是合法且合乎标准的 CSS，在各种浏览器中都能正常工作。

这三种语法的原理过于复杂，所以这里就不解释了，但是这确实也不太重要。三种中任意一种都能满足我们的目的，所以选择哪一种真的只和个人喜好有关。我选择的是“Bulletproof Smiley”版本。

说明　如果你想知道更多细节，请点击 Font Squirrel Generator 里每个 CSS 格式名下方的蓝色链接来阅读相关的三篇文章。

下面就是 Prelude 字体的 Bulletproof Smiley 语法：

```
@font-face {
    font-family: 'Prelude';
    src: url('fonts/preludeflf-webfont.eot');
    src: local('☺'),
      url('fonts/preludeflf-webfont.woff') format('woff'),
      url('fonts/preludeflf-webfont.ttf')
      ¬ format('truetype'),
      url('fonts/preludeflf-webfont.svg#webfont')
      ¬ format('svg');
}
@font-face {
    font-family: 'Prelude';
    src: url('fonts/preludeflf-bold-webfont.eot');
    src: local('☺'),
      url('fonts/preludeflf-bold-webfont.woff')
      ¬ format('woff'),
      url('fonts/preludeflf-bold-webfont.ttf')
      ¬ format('truetype'),
      url('fonts/preludeflf-bold-webfont.svg#webfont')
      ¬ format('svg');
    font-weight: bold;
}
```

请将这段语法置于所有 CSS 规则之前，其实放在任何位置，它都能正常工作，但是接下来你会了解到，将它置于你的样式之前的话，会如何提高页面的性能。你可以直接在练习文件的 paper_final.html 中复制和粘贴这段代码。

这两段`@font-face`规则通过声明同样的`font-family`名字，Prelude，将正常和粗体两种字体组合成一个字体族。每个`@font-face`规则都给出了字体文件的路径，并且还有一些样式和特性（比如`font-weight: bold`和`font-style: italic`）。

说明　在 Paul Irish 的原创文章（http://paulirish.com/2009/bulletproof-font-face-implementation-syntax）中细述了这些语法的关键细节。这些知识不必掌握，但如果你和我一样是个 Web 极客，我想你会对它很感兴趣。

我们先看看第一段`@font-face`声明，然后一行行地读下来。

规则的第一部分——`font-family: 'prelude';`——给你链入的字体定义了一个名字，这样之后你可以在字体定义的字体栈里面声明对它的引用。你可以随意命名，它仅仅是对整个字体信息的一个简单引用。

规则的第二部分——`src: url('fonts/preludeflf-webfont.eot');`——为 IE 8 及之前的版本指明了 EOT 版本的字体文件路径。这个声明和其他版本的字体文件声明分开的原因是因为 IE 浏览器无法理解在`src`描述符里面使用逗号来分割的多个值。它会把那看作一个路径，并将使得 EOT 格式的文件没法在 IE 下起作用。

规则的下一部分是第二个 `src` 值，这次列出了针对非 IE 浏览器的所有字体文件。每个浏览器都会扫描这个字体文件列表，直到它找到一个合适的字体文件为止，然后它就会下载文件，也只有那个文件，会显示成文本。每个字体文件声明都会包含一个字体文件的路径，比如 `url('fonts/preludeflf-webfont.woff')`，以及格式的提示，比如 `format('woff')`。格式提示是可选的，但包含这个的话能够提醒浏览器每个字体文件的具体格式，也能避免浏览器下载不能使用的文件，这种下载将会严重浪费带宽并降低页面速度。

节省服务器资源

所有的浏览器，除了 IE 8 及之前的版本之外，在 CSS 规则的字体定义中真正使用这个字体之前，是不会做任何下载的。这意味着你可以在样式表中声明大量的`@font-face` 规则，只要你的页面中并没有什么地方实际使用到这些字体，浏览器就不会发起一个额外的 HTTP 请求下载字体。

相反地，IE 8 及之前的版本只要遇到 EOT 文件的声明就会开始下载它。当你测试字体嵌入的时候，通常会链入大量的字体文件以便观察哪种字体的效果最合适。请确保发布时移除那些你并不需要的字体文件，以保证 IE 8 及之前的版本不会下载没必要的 EOT 文件。

说明 WOFF 格式被写在最前面是因为它是标准，而我们希望如果浏览器支持则会首先使用它。它同样也是尺寸最小的格式，因此最好让浏览器首先识别并使用它。

你可能会注意到，第二个 `src` 的开始值是 `local('☺')`，这个笑脸表情到底有什么用？

这个笑脸表情放在第二个 `src` 的开始处，目的是为了保护 IE 浏览器。如果没有它的话，IE 会尝试把第二个 `src` 描述符的值读作一个路径。这我们之前已经解释过了，最终它会导致一个 404 错误。虽然这不会让`@font-face` 停止工作——IE 还是能够使用独立定义的 EOT 文件——但是这是对你的服务器发起的一个额外的、无用的请求，你显然不希望这种事情发生。IE 无法理解 `local()`语法，所以将它放在 `src` 的最前面，就可以保证 IE 会完全忽略掉这条定义。

`Local()`的问题

让用户避免下载已经安装过的字体听起来是个好主意——所以为什么不在 `local()`中使用真实的字体名，而要使用笑脸表情呢？这确实只是一个可选项。Paul Irish 的“bulletproof `@font-face` syntax”是这么做的，Font Squirrel 生成器也是这么做的。当你在 `local()`中使用真实的字体名之前，应该意识到以下的几个问题。

- 不同的字体可能有同样的名字。所以很有可能产生重名问题，这个可能性很小，但是是有可能的，参见 http://typophile.com/node/63992。
- 如果用户机器上的字体是用 FontExplorer 来安装的话，在 Chrome 上会遇到奇怪的问题，参见 http://snook.ca/archives/html_and_css/font-face-in-chrome。

> 如果用户机器上的字体是用 FontExplorer 来安装的话，在 Safari 上系统可能会弹出对话框来向用户要求使用本地字体的权限。
>
> 这些问题并不总是发生，但是如果它们发生了，影响会非常坏。许多 Web 字体专家建议安全不要使用 `local()`。除非你希望阻止用户下载的字体文件非常大。

顺便说一下，`local()`语法是合法的 CSS。它的真正用途是在`@font-face`规则中指向用户机器上本地安装的字体版本，因此如果用户已经安装了和你嵌入的字体一样的版本，那他就无需额外下载文件了。这就是为什么贡献这段语法的 Paul Irish 建议在这里使用一个笑脸符的原因：我们并不希望有可能指向一个真实存在的文件，而应该不太可能有人会发布一个字体，并将其命名为☺。

第二段`@font-face`规则声明了粗体版本的 Prelude。它声明了所有粗体字体文件的路径并且将`font-weight`设置成`bold`。但是`font-family`的名字还是 Prelude（不是 PreludeBold 或者其他什么名字），跟第一段的名字匹配。使用同样的名字可以确切地告诉浏览器这个文件是同样的 Prelude 字体族的粗体版本。任何时候浏览器需要加粗的 Prelude 文本时（比如 HTML 中使用了`strong`元素或者 CSS 有`font-weight: bold`定义），它不会去对常规的字体做任何加粗的工作，而是会直接使用对应的粗体字体文件。使用真实的粗体文件比浏览器模拟效果要好多了。

说明　IE 并不总能正确地处理`@font-face`规则里的字体样式。当`@font-face`规则里有`font-style: italic`时，IE 8 及之前的版本并不使用指定的字体。IE 9 会使用，但它强制渲染成斜体样式，即使你所指定的字体并不是斜体。

---

2. 声明字体

在 CSS 中添加`@font-face`并不会实际上使用某种字体，它只是简单链接了这些字体，使你在需要使用它们的时候可以下载并使用。让我们试试在页面的`h1`和`h2`元素中使用这种字体。使用`@font-face`中所指明的字体名字“Prelude”，把它添加到`h1`和`h2`的已有的`font-family`的值的最前面：

```
h1 {
    margin: -.3em 0 .14em 0;
    color: #414141;
    font-family: Prelude, Helvetica, "Helvetica Neue",
                 Arial, sans-serif;
    font-size: 3.5em;
    font-weight: normal;
}
h2 {
    clear: left;
    margin: 0 0 -.14em 0;
    color: #414141;
    font-family: Prelude, Helvetica, "Helvetica Neue",
```

```
                 Arial, sans-serif;
     font-size: 2.17em;
     font-weight: bold;
}
```

字体栈中的备选字体 san-serif 看起来和手写的 Prelude 字体完全不同。我选择这些备选字体只是因为实际上目前根本没有任何手写的 Web 安全字体可供我们用作备选。如果某人的浏览器不能够支持字体嵌入，那我宁愿他看到的是清晰干净的 Htlvetica 或者 Arial 字体，而不是机器上安装的某种手写字体。

请注意 `h1` 规则将 `font-weight` 设置成普通而 `h2` 规则将 `font-weight` 设置成粗体。这告诉浏览器针对 `h1` 元素使用正常的 Prelude 字体(第一段规则),针对 `h2` 元素则使用粗体的 Prelude 字体（第二段规则)。最终效果如图 3-29 所示。



图 3-29 标题中的手写 Prelude 字体

我们现在拥有了可以缩放、可以选择也可以索引的手写文本。在不同的浏览器中以及在 Windows 和 Mac 系统中存在不同的文本抗锯齿和精调效果，但是使用真实文本的优点足以让我们接受某些浏览器中可能存在的轻微锯齿所带来的不一致。(图 3-30)

表3-6 `@font-face`的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持	支持，3.5+	支持，10+	支持	支持



图 3-30　不同浏览器的抗锯齿效果不同，左边是 Firefox 3.6，右边是 IE 9

`@font-face` 规则的来龙去脉

字体精调和抗锯齿是一个远远超越本书范围的技术问题，如果你想了解更多的话，可以看看下面的一些文章。

`@font-face` 是字体（Fonts）模块的一部分，参见 www.w3.org/TR/css3-fonts。

`@font-face` 规则需要给出一个字体的名字（使用 `font-family` 描述符），这个名字可以任意指定，同时还要给出一个到多个指向字体文件的路径（使用 `src` 描述符）。它也可以包括一些样式特性（使用 `font-weight`、`font-style` 和 `font-stretch`）。你可以通过指定同样的字体名让多个字体文件组合成一个字体族。

如果要使用`@font-face` 规则中链入的字体文件，只需要简单地在 `font-family` 属性里面的字体栈加入之前声明的字体族的名字。

除了我们例子中的手写字体以外，`@font-face` 特性还可以用在以下一些场合。

- 创作一种使用 Web 安全字体无法实现的外观。
- 保证网站的商标和 logo 等能够和印刷物上的一致。
- 显示非拉丁语系的字符，它们通常在浏览器的默认字体下比较难看。使用专为这种语言设计的字体能够保证所有的字符都正确显示。

还有一种使用`@font-face` 的尝试是使用符号字体来创建不需要图片的图标。但是这种方法有非常严重的可访问性问题。参见 http://filamentgroup.com/lab/dingbat_webfonts_accessibility_issues 以及 http://jontangerine.com/log/2010/08/web-fonts-dingbats-icons-andunicode，这里有关于这个问题的讨论以及潜在的解决方案。

3. 提高性能

你如果现在在浏览器里面查看这个练习页面，可能会注意到在页面加载完成和手写字体显示之间会有一个滞后时间。在字体文件下载完毕之前，基于 Webkit 的浏览器无法显示`@font-face` 定义的字体效果（见图 3-31）。

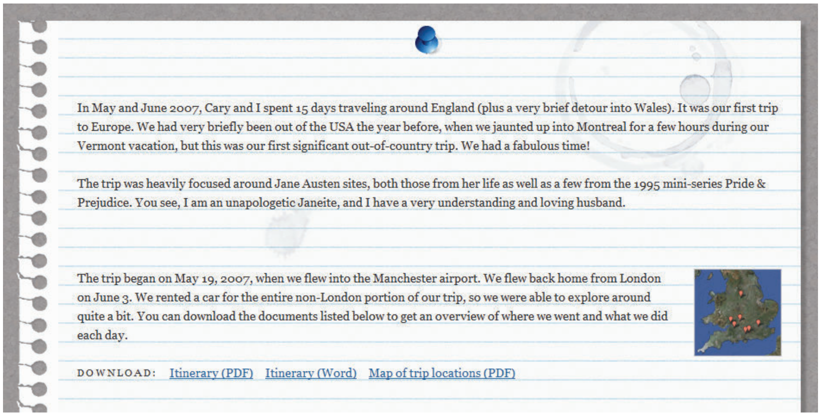


图 3-31 字体文件下载之前无法显示

在 Firefox 和 Opera 上，备选的字体将会显示一会儿，直到字体文件下载完成，然后浏览器就会使用新字体重新渲染文本。这被称为 Flash of Unstyled Text，或者 FOUT，是由 Paul Irish 创造的一个词。

这些字体加载的滞后问题通常来说不会很严重，但是在某些场合下还是很容易被人注意到。非西方语言的字体，比如中文或者日文，包含了成千上万的字符，大小通常会按兆来计算。这些巨大的字体文件当然会花费很长的时间下载。当持有移动设备的用户处于信号覆盖不好的区域，或者在网速非常慢的宾馆时，这些情形下都要花非常长的时间等待字体下载。

说明 之前所提到的 `font-size-adjust` 属性并不会缩短 FOUT 的过程，但由于它可以使备用字体的尺寸贴合 Web 字体，所以可以使整个过程变得平滑。但需要再一次提醒的是，仅有 Firefox 支持它。

有很多种方法可以减少这种等待，处理 FOUT 问题，或者说 Webkit 上文本不可见的问题。

- 保持你的字体文件尽可能小，尽可能的减少字体文件中的字符，做到只包含需要的字符当然是最好。Font Squirrel 的生成器就能够帮我们做到这一点。
- 把`@font-face` 规则放在 CSS 的最上方。这会增加字体文件优先加载的机率，比如它就可能在背景图片之前加载。
- 让浏览器尽快启动字体文件的下载，比如在页面开始处的隐藏元素里面就请求对字体的使用。有很多图片预取技术可以参考运用到字体文件上，参见这里 http://perishablepress.com/press/tag/preloadimages。
- 扩大字体的伺服范围。如果在一个比较常用的地方提供字体服务的话，你就会增加这样一种几率，访客的缓存里面可能早就存在这种字体了，所以就不必再从一个私有的地址

上再次下载了。之前列出来的字体嵌入服务能够帮助你做到这一点，比如说 Google 的字体目录服务。你甚至可以上传你的字体到 TypeFront 服务上去（http://typefront.com）。TypeFront 可以伺服你上传的字体，将它们转换成各种所需的格式，还能限制只对哪几个网站提供这种字体的服务。

- 在.htaccess 里面将超时设得很长，这样能保证字体文件只被下载一次。浏览器会缓存这个字体文件，并且在很长一段时间内不再发起请求。这对页面的第一次访问没有任何帮助，但是对后续的请求还是很有好处的（参见 Steve Lamm 的“HTTP 缓存”http://code.google.com/speed/articles/caching.html）。
- 用 Gzip 压缩你的字体文件。Stoyang Stefanov 发现 Gzip 之后平均文件大小将会减少百分之 40 到 45（参见 www.phpied.com/gzip-your-fontface-files）。但是他同时也发现这对 WOFF 字体文件没有太大的帮助，因为这种格式已经有较高的压缩率了，因此这可能对 Firefox 中存在的 FOUT 问题没有多大的改善（参见 www.phpied.com/font-facegzipping-take-ii）。但 Gzip 还是能够帮助 Opera 改善 FOUT 问题，也能帮助 Safari 和 Chrome 更快地显示文本。
- 在浏览器下载字体文件期间，使用脚本来隐藏页面上的所有内容。这对字体下载的实际速度没有任何意义，但是避免用户看到 FOUT 问题的整个变化过程。Paul Irish 提供了两个脚本帮助我们这么做，其中一个使用了 Google 的 Web 字体 JavaScript 加载库（http://paulirish.com/2009/fighting-the-font-face-fout）。

我们的字体文件已经使用了最小子集，并且放在了 CSS 文件的最开始处，因此基本上覆盖了`@font-face`性能的最佳实践。对页面做脚本或者服务器端的优化已经超越了本书所涵盖的内容，但当你在 Web 字体的加载时间方面遇到问题时，不妨试试上面这些办法。

3.4　大功告成

我们已经完成了对这个文章页所做的样式工作，现在它已经真的像一页纸质笔记了。在一个现代的非 IE 浏览器里面，你应该能够看到如图 3-32 所示的效果，可以拿它跟图 3-1 比较一下。

IE 9 的预览版没法显示边缘的撕纸效果，但是看起来跟图 3-32 基本上差不多了。IE 8 及之前的版本缺失了大部分图像效果，但是它们也能显示线格背景图片和手写字体，总体表现还是很有吸引力，也比较像纸质笔记页（见图 3-33）。与此同时，这个例子里面所有 IE 8 及以前版本显示都非常一致，甚至 IE 5.5 也能显示图 3-33 的效果。

说明　完成版页面所展示的效果代码都在`paper_final.html`页面里，它在你下载的本章的示例文件中。



图 3-32　最终页面的效果（另见彩插图 3-32）



图 3-33　IE（最高到版本 8）没法显示所有的图像细节，但是能够显示手写字体

第 4 章

根据类型为图片和链接定义样式

在网络上你很难找到一个没有链接和图片的页面。除了指向其他页面之外，这些链接还可能指向一些不同类型的文档，比如 PDF 或视频文件。而那些图片则可以分为不同的类型，如照片、插图或图表。使用 CSS3 可以很方便地根据图片和链接本身的独特类型来定义样式，但也需要花费更多的时间和精力，需要更复杂的结构。你可以借助 CSS3 中新的选择器以一种更特别、高效的方式分别定位不同类型的链接和图片。值得再次重申的是，CSS3 的出现可以同时提升你的开发效率和页面的执行效率。

你将学到

我们将根据链接的类型为其增加图标，并单独为照片型的图片定义不同的样式，这需要用到如下 CSS3 选择器：

- "值末匹配"（End of the Value）（$）属性选择器；
- "值中包含"（Somewhere in the Value）（*）属性选择器。

4.1 基础页面

这次我们将要使用的起始页面是第 3 章的完成版页面（见图 4-1）。它包含了大量不同类型的链接，然而目前所有的这些链接都是相同的样式。如果能为特定的文件链接定义不同的样式，体验应该会很不错，这样就能在用户打开文档之前给他一个视觉提示。该页面同样包含若干图片，它们多数都是照片，仅有一幅是地图的缩略图。同样地，如果能为照片单独定义样式，效果也会很好，但至今为止所有的图片都仍拥有同样的样式。

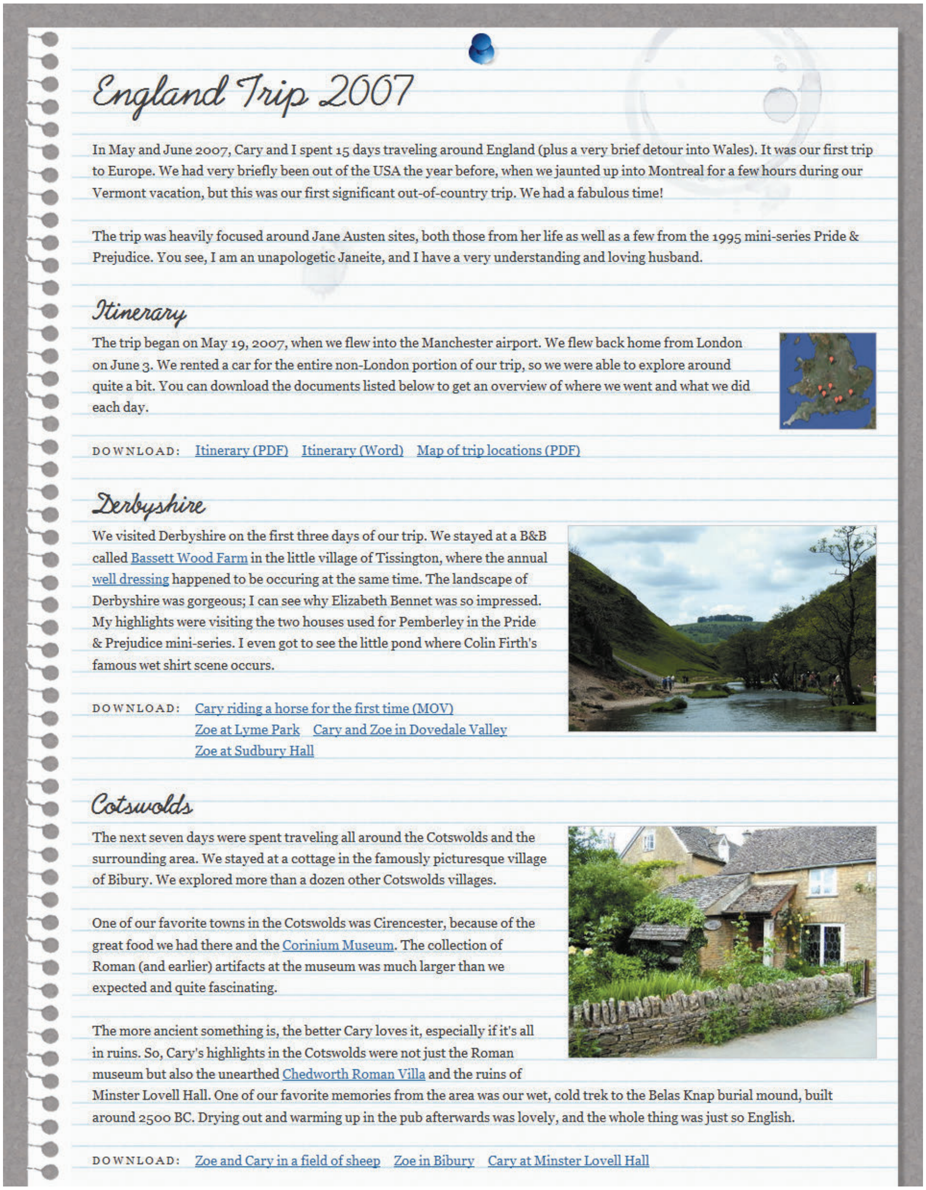


图 4-1　所有的链接都是相同的样式，图片也是一样

4.2　什么是属性选择器

我们可以使用**属性选择器**（Attribute Selectors）来针对不同类型的链接和图片定义不同的样式。属性选择器既强大又实用，借助它你可以定位到 HTML 里的某个元素却无需知晓其 ID 或

类名。属性选择器是根据元素的某个特定属性是否存在或其属性值是否符合匹配条件来定位到它的。

说明 一个属性选择器并非只能与类型选择器（Type Selectors）关联起来使用。你可以将它与任何简单选择器（Simple Selector）组合使用。比如说`.warning[title]`就结合了一个类选择器（Class Selector）和一个属性选择器。你也可以单独使用它，`[title]`将选择每个拥有 `title` 属性的元素。

举例来讲，`img[alt]`是由一个类型选择器 `img` 和一个紧随其后的属性选择器`[alt]`组合而成。所有的属性选择器都被方括号包围着，括号中的内容则根据你想要选择的内容而定。`img[alt]`这个选择器的目标是所有设置了 `alt` 属性的 `img` 元素。在你的测试页面使用这个选择器，可以给所有拥有 `alt` 属性的图片定义一个亮绿色的外框，这样一来就可以一目了然地知道哪些没有外框的图片需要补上 `alt` 属性。

```
img[alt] {
    border: 3px solid #0C0;
}
<img src="photos/poe.jpg" width="320" height="241"
¬alt="My cat Poe">
<img src="photos/dog.jpg" width="320" height="240">
```




图 4-2 猫咪图片的 `img` 元素拥有 `alt` 属性，因此它有绿色外框。但狗的图片却没有

`img[alt]`是属性选择器最简单的示例之一——它只检查某个属性是否存在，但与其属性取值无关。它也是 CSS 2.1 规范所包含的 4 个属性选择器（见表 4-1）中的一个。

表4-1　CSS 2.1属性选择器

属性选择器	功　能
[attr]	匹配拥有attr属性的元素，与属性的取值无关
[attr=val]	匹配拥有attr属性的元素，其属性值必须刚好等于val
[attr~=val]	匹配拥有attr属性的元素，其属性值是用空格分隔开来的单词列表，其中之一必须刚好等于val
[attr|=val]	匹配拥有attr属性的元素，其属性值必须刚好等于val，或者以val开始并紧跟着一个连字符

W3C 将这些 CSS 2.1 的属性选择器称作*属性存在性及值（匹配）选择器*（Attribute Presence and Value Selectors）。

CSS3 引入了 3 个全新的属性选择器，它们可以为你想要匹配的目标提供更为精确的控制。

表4-2　CSS3属性选择器

属性选择器	功　能
[attr^=val]	匹配拥有attr属性的元素，其属性值必须以val开始
[attr$=val]	匹配拥有attr属性的元素，其属性值必须以val结束
[attr*=val]	匹配拥有attr属性的元素，其属性值必须包含val

提示　如果你在书写一个复杂的选择器时感到困惑，可以将它贴入 SelectORacle（http://gallery.theopalgroup.com/selectoracle），它将用平实的英语（或西班牙语）讲解这个选择器将匹配哪些目标元素。

W3C 将这些 CSS3 的属性选择器称作*子（字符）串匹配属性选择器*（Substring Matching Attribute Selectors），因为它只匹配属性值的一部分而不是全部。

属性选择器配合示例才最容易理解，所以现在我们把它们加到页面里来看看效果，并了解一下实际用起来到底如何。

不　匹　配

一个名为“不匹配”的属性选择器可能是相当实用的，你可以用它来选择所有不匹配给定条件的元素。举例说，为了对非提交按钮的 input 元素定义样式，你可以用它来创建这样一条规则——“找到所有 type 属性值不为 submit 的 input 元素”。

遗憾的是，目前并不存在这样一个属性选择器，但有一个方法可以模拟实现：使用“:not 选择器”，它是 CSS3 的一个新伪类。它选择与你指定的条件刚好相反的那些元素。因此，input:not([type=submit])这样一个选择器便会告诉浏览器“找到所有 type 属性值不为 submit 的 input 元素”。:not 选择器兼容于除 IE 8 及之前版本之外的所有主流浏览器。你可以从 http://kilianvalkhof.com/2008/css-xhtml/the-css3-not-selector 获得更多有关它的信息。

如果你正在使用 JavaScript 框架，它可能已经包含了某种“不匹配”属性选择器；请参阅 jQuery（http://api.jquery.com/attribute-not-equal-selector）和 MooTools（http://mootools.net/docs/core/utilities/selectors）的选择器相关文档。

4.3　用动态添加的图标指示文件类型

开始之前，请到 www.stunningcss3.com 下载本章的示例文件并用编辑器打开 selectors_start.html 页面。其 CSS 的定义包含于页面 `head` 里的一个 `style` 元素里。该页面和第 3 章的文件相同，你也可以将第 3 章的最终页面作为本章的起始点。

纵观整个页面，有链接对如下类型的文档提供下载：

- PDF
- MOV
- DOC
- JPG

某些情况下，我会将文件类型括起来置于链接文字的末尾来提示用户，比如“旅游路线图（PDF）”。我希望告知用户当他点击某些链接时可能启动额外的应用程序（例如 Adobe 公司的 PDF 阅读器 Acrobat Reader），或是弹出下载询问框。但作为一个普通的人类，我可能无法保证在每个类似的链接后都加上文件类型的提示。又或者该页面是 CMS 程序的一部分，由我那不谙技术的客户来负责增加链接，现在怎么办？他甚至可能不愿去增加这个友好的文件类型提示。

一个更加万无一失的办法便是使用属性选择器来增加文件类型的提示，而这一切将由浏览器自动完成。所有链接的末尾都有文件类型后缀，因此我们使用“值末匹配”（End of the Value）这个属性选择器来检查文件后缀，并为其 `a` 元素增加与之相对应的图标背景。

首先，在增加背景图之前先给文件下载列表里的 `a` 元素做一些准备工作：

```
ul a {
    display: block;
    min-height: 15px;
    padding-left: 20px;
    background-repeat: no-repeat;
    background-position: 0 3px;
}
```

这段定义会使链接变成块元素，并拥有与图标高度相符的最小高度值，这样一来图标就不会显示不全。它还增加了左内边距，为了给图标腾出显示的空间。每个图标的背景图片将仅显示一次（`no-repeat`），而且比链接文字的顶部低 3 个像素（`0 3px`）。

何时需要加上引号

只有当选择器的值是字符串时才需要加上引号。如果值是标识符,就不必为其加上引号，但即使加上也无妨。标识符与字符串的区别基本上在于标识符限定了它可以包含或作为其前缀的字符。关于标识符的定义，请参阅 www.w3.org/tr/css21/syndata.html#value-def-identifier；而字符串的定义则请参阅 www.w3.org/tr/css21/syndata.html#strings。

如果你不想费脑筋记忆这些区别，安全起见你可以给属性选择器的值统统加上引号。

现在我们可以增加属性选择器来匹配每个文件类型的后缀了：

```
a[href$=".pdf"] {
   background-image: url(images/icon_pdf.png);
}
a[href$=".doc"] {
   background-image: url(images/icon_doc.png);
}
a[href$=".mov"] {
   background-image: url(images/icon_film.png);
}
a[href$=".jpg"] {
   background-image: url(images/icon_photo.png);
}
```

说明 这些图标文件来自由 Mark James 设计的 famfamfam Silk 免费图标集（www.famfamfam.com/lab/icons/silk）。

每个属性选择器里的 `href$=`告诉浏览器要“找到所有以某个值结尾的 `href` 属性值”，而接下来那些被引号括起来的值——比如 `.pdf`——就是用来匹配的参考标准。当浏览器找到合适的元素时，便对它们应用背景图的样式，从而为链接添加相应的图标（见图 4-3）。

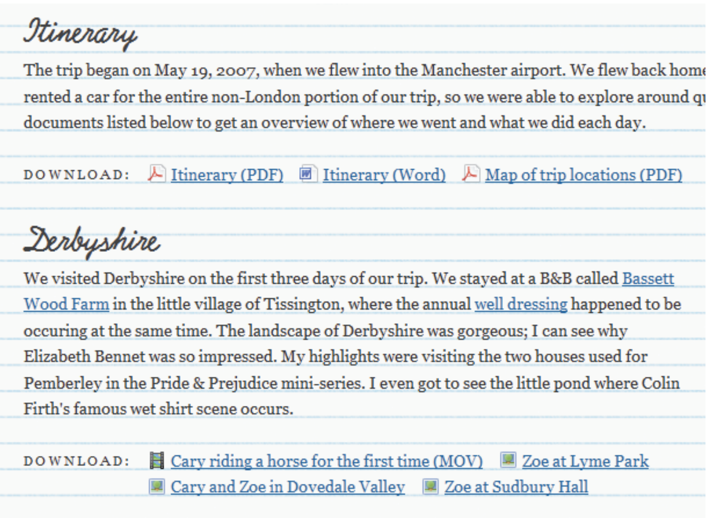


图 4-3 现在每个链接的旁边都有一个与其文件类型相对应的图标了

说明 至此为止的所有修改都在 `selectors_1.html` 页面里，它在你所下载的本章的示例文件之中。

图标的缩放

除了对链接设置 min-height 来确保背景图标不会被截断显示以外，你还可以用 background-size 来让图标与文字进行等比缩放。相应的规则应该看起来如下所示：

```
ul a[href] {
   display: block;
   padding-left: 20px;
   background-repeat: no-repeat;
   background-position: 0 3px;
   -moz-background-size: 1.2em;
   -webkit-background-size: 1.2em;
   background-size: 1.2em;
}
```

但是，在示例页面里我并没有加入这段代码，因为当浏览器对图标进行缩放时，它们会变得模糊——即使是将图标缩小而不是拉伸。对于线条清晰的图标，特别是这些小图标，我并不建议让浏览器本身对其进行缩放显示。但请记住这一条技巧，当你下次遇到边缘不太清晰的图片时，因缩放带来的显示模糊将不会那么显眼。

4.3.1 除图标之外的备选方案

我们已经完成了对链接图标的样式定义，如果你愿意，还可以将属性选择器使用在其他的场景。

1. 放弃图片而改用文字描述

图标可以很好地起到帮助并提示用户的作用，但是如果你希望提示显得更醒目和明确，你可以选择使用“内容生成”特性在链接的后面增加文件类型的后缀名，它可以用来替换图标或是作为图标的补充。

你首先需要确保这些文件的后缀名信息并没有被你手工添加到链接上。接着，你需要添加以下的样式规则，举例而言，要为每个 PDF 文件的链接之后加上“(PDF)”字样，代码为：

```
a[href$=".pdf"]:after {
    content: " (PDF)";
}
```

2. 组合多个属性选择器

和其他类型的选择器相同，你可以将多个选择器组成一个来方便你更精确地控制想要匹配的元素。比如，如果你想为指向 PNG 图片的链接显示相片图标，但恰好需要显示图表图标的图片也是 PNG 格式的，你该怎么办呢？以下的选择器便可以解决这一问题，当然它取决于图片的命名：

```
a[href$=".png"][href*="chart"] {
    background-image: url(images/icon_chart.png);
}
```

这个选择器告诉浏览器“找到所有 href 的属性值以‘.png’结尾且包含‘chart’字样的元素”。这样一来，以下所有的链接都将被匹配：

```
<a href="images/chart_locations.png">
```

```
<a href="images/piechart.png">
<a href="charts/travel.png">
```

4.3.2　修正IE 6 的问题

IE 6 是主流浏览器中唯一不支持属性选择器的，因此将不会显示那些图标。仅有的解决方案便是加载脚本来提供对属性选择器的支持。

1. 增加针对 IE 的修正脚本

Dean Edwards 编写的一个脚本能够使你代码里的这些高级选择器成功地发生功用，但令人纳闷的是这个脚本名叫“IE 7”（http://code.google.com/p/IE 7-js）。你可以下载这个脚本并放进你的网站进行本地调用，也可以直接引用 Google Code 上的公共版本。引用公共版本的好处是：访客在浏览其他网站时已经加载过这个脚本，那么在访问你的页面时将从缓存里直接读取，这样会使页面加载得更快。

将公共版本的脚本引入页面的 head 中，并加上 IE 6 才能识别的条件注释：

```
<!--[if IE 6]>
<script src="http://IE 7-js.googlecode.com/svn/
version/2.1(beta4)/IE 7.js"></script>
<![endif]-->
```

该脚本将使 IE 6 也读得懂属性选择器并让图标显示出来，但它会引发一个奇怪的问题，即链接本身会出现空白并折行。为了修复这个问题，我们要使链接变成 inline-block 并将 white-space 设置为 nowrap，但这又会引发其他浏览器的一些小问题。因此，我们需要对 html 标签加上与我们在第 2 章里所使用的相同的条件注释来创建只有 IE 6 才能识别的规则。

找到页面起始处的 html 标签，将它更改成如下 HTML 代码：

```
<!--[if lt IE 7 ]><html lang="en" class="ie6"><![endif]-->
<!--[if IE 7 ]><html lang="en" class="ie7"><![endif]-->
<!--[if IE 8 ]><html lang="en" class="ie8"><![endif]-->
<!--[if IE 9 ]><html lang="en" class="ie9"><![endif]-->
<!--[if gt IE 9]><html lang="en"><![endif]-->
<!--[if !IE]>--><html lang="en"><!--<![endif]-->
```

提示　如果不想手动输入上述代码，请在本章的示例文件里找到并打开 selectors_final.html 页面，将这部分代码复制粘贴到你的页面。

现在你就可以增加只有 IE 6 才能识别的代码了：

```
.ie6 ul a {
    display: inline-block;
    white-space: nowrap;
}
```

IE 6 现在和其他浏览器一样能够正常显示了——前提是 JavaScript 功能已启用。如果某个 IE 6 用户禁用 JavaScript，那么他将看不到那些图片。这没有问题——这些图标只是功能上的改进，而并非必要内容。但禁用 JavaScript 以后，IE 6 将依旧读取 ul a 这条规则，它会给每个链接增加额

外的内边距，从而使 IE 6 上的链接之间出现不必要的空白。为了解决这个问题，请为 ul a 和.IE 6 ul a 两条规则都加上如下的[href]属性选择器：

```
ul a[href] {
   display: block;
   min-height: 15px;
   padding-left: 20px;
   background-repeat: no-repeat;
   background-position: 0 3px;
}
.ie6 ul a[href] {
   display: inline-block;
   white-space: nowrap;
}
```

这些规则匹配的目标是所有 ul 元素下拥有 href 属性的 a 元素，这些规则与改动前的规则所匹配的链接完全相同，然而当 JavaScript 被禁用时，IE 6 便无法识别属性选择器，因此它将完全忽略以上两条规则，也就避免产生额外的内边距或其他因此而增加的样式。

说明　至此为止的所有修改都在 selectors_2.html 页面里，它在你所下载的本章的示例文件之中。

---

2. 使用 JavaScript 库

除了已经成功使用的 IE 7 脚本之外，我们还有另外的备选方案，即选用内置已兼容了属性选择器的 JavaScript 库或框架，然后在你的代码中引入它们并完成想要的效果。本方案的缺点是它不会使你已经编写好的 CSS 代码自行生效，需要编写脚本来重新实现相同的效果。然而，假如你已经编写了一段脚本来实现页面中的某些效果，那么更好的方案便是为其附上选择器，而不是加载 IE 7 脚本来实现兼容。

"Selecting and Styling External Links, PDFs, PPTs, and other links by file extension using jQuery"（使用 jQuery 根据文件后缀来选取并定义外链、PDF、PPT 和其他链接的样式）（http://dabrook.org/blog/articles/selecting-and-styling-external-links-or-pdf-ppts-and-other-files -by-extensi）一文用示例介绍了如何使用 jQuery 所支持的选择器为链接增加图标。大多数主流的 JavaScript 库，比如 jQuery 和 MooTools，都内置了选择器功能，而且还有少数库专攻选择器这一功能，它们包括：

- YUI Selector Utility (http://developer.yahoo.com/yui/selector)；
- Sizzle (http://sizzlejs.com)；
- Sly (http://github.com/digitarald/sly)。

还有一些脚本借助某个 JavaScript 库所包含的选择器功能帮助你省掉重新编写脚本的工作——它们直接探测你在 CSS 中已经编写好的选择器，然后使之生效。我们在第 1 章介绍过的由 Keith Clark 编写的 Selectivizr 脚本（http://selectivizr.com）是一个不错的选择。先将该脚本直接引入到你的页面中，再从 7 个 JavaScript 库中选择一个引入，这样一来属性选择器就能够在 IE 上工作了。如果你正在使用 jQuery，还有一个选择便是使用名为 SuperSelectors 的插件（http://github.com/chrispatterson/jquery-super-selectors）。

4.4 区别处理全幅图和缩略图的样式

页面中另一个非常棒的应用是使用属性选择器分别为照片和地图缩略图定义不同的样式。如果抛开 CSS3 来实现这个需求，我们可以简单地给缩略图加上一个类，并为这个类应用一个独特的样式。对于本页面来说或许非常容易实现，但在实际开发中使用类并不是最简便的方法。

4.4.1 使用类可能遇到的问题

尽管类有很多合理的使用场景，但某些情形下仍然会出现一些问题并影响我们的使用。

- **类会增加 HTML 代码量**。在我们的例子中，增加一个类并不会有什么太大影响，但在那些样式更为复杂的页面或网站里，大量的类便会大幅增加文件尺寸。无论什么时候，只要你能避免增加类或 ID 来定位元素，请毫不犹豫地选择其他方法。
- **页面结构可能受控于 CMS 或插件**，这可能使你无法往页面中增加类。
- **你的客户也是内容的编辑之一**，你不能强求他去指定合适的类。
- **或许你根本无权编辑 HTML**，因为你在项目里担当着 CSS 开发者的角色，也可能你的任务只是对某个现有项目的 CSS 进行一些风格上的更新。
- **增加类会花费大量的时间**，假如你正准备给一个拥有许多页面的网站增加一些新样式。更简单的方法是利用现有的 HTML 编写 CSS，而不是为了新增样式修改页面的结构。

容错措施

尽管页面的开发者有可能会将图片放错文件夹，但相较之下我倒觉得某个客户把类名称写错的可能性会更大。如果你想为此增加一层额外的保障，可以为其多指定一个类，然后为这个类和属性选择器同时指定相同的样式。这样一来，如果某个人忘记指定类，则属性选择器就会生效；而假如另一个人将文件放错了地方，那么类的样式就会发生作用。这属于额外的工作，但却是一种保障措施，当你忘记属性或类中任意一个时便会自动纠正错误。而且，当你忘记为 IE 6 使用兼容脚本时，类也会有其功用。

如果你将一个类选择器和一个属性选择器分组到一起，请注意 IE 6 将会忽略整条规则，甚至它理应读取并使用类选择器的那部分样式（当然，如果你用脚本为 IE 6 增加了对属性选择器的支持，则无此问题）。为了解决这个问题，你必须将规则分开书写，就像这样：

```
img[src*=thumbnails] {
    float: left;
    margin: 0 20px 10px 0;
}
img.thumbnail {
    float: left;
    margin: 0 20px 10px 0;
}
```

代码存在冗余，但如果你不希望借助脚本就能兼容 IE 6，那么这是你唯一的选择。然而我们不会在示例文件里这样做，并不是仅仅因为我们使用了脚本实现兼容，而是因为我们用属性选择器实现了并不重要的装饰性效果，我认为 IE 6 丢失这些效果也无伤大雅。但如果你要用它实现更重要的效果，或是用来提供容错，那么上面的方案可供你选择。

属性选择器的来龙去脉

属性选择器是选择器（Selector）模块的一部分，你可以在 www.w3.org/tr/css3-selectors 找到它。以下是 7 个属性选择器。

- `[attr]`匹配拥有 `attr` 属性的元素，与属性的取值无关。
- `[attr=val]`匹配拥有 `attr` 属性的元素，其属性值必须刚好等于 `val`。
- `[attr~=val]`匹配拥有 `attr` 属性的元素，其属性值是用空格分隔开来的单词列表，其中之一必须刚好等于 `val`。
- `[attr|=val]`匹配拥有 `attr` 属性的元素，其属性值必须刚好等于 `val`，或者以 `val` 开始并紧跟着一个连字符。
- `[attr^=val]`匹配拥有 `attr` 属性的元素，其属性值必须以 `val` 开始。
- `[attr$=val]`匹配拥有 `attr` 属性的元素，其属性值必须以 `val` 结束。
- `[attr*=val]`匹配拥有 `attr` 属性的元素，其属性值必须包含 `val`。

前 4 个是 CSS 2.1 的一部分，被称为属性存在性及值（匹配）选择器（Attribute Presence and Value Selectors）；后 3 个则被称为子（字符）串匹配属性选择器（Substring Matching Attribute Selectors），属于 CSS3。

属性选择器里的值可以是标识符或字符串，字符串必须用引号括起来。

属性选择器的特征与类和伪类选择器的特征相同。

除了链接图标和文件类型图片等需求以外，你还可以将属性选择器用在以下场景。

- 为不同的表单域设置独特的样式（比如使用 `input[type=submit]`），请参阅 http://dev.opera.com/ articles/view/styling-forms-with-attribute-selectors。
- 为不同的语言指定不同样式（比如使用`[lang|=en]`）。
- 为拥有 `title` 属性的元素增加视觉提示（使用`[title]`）。
- 为导航型 `div` 元素里的列表隐藏“小圆点”（Bullet）（比如用 `div[id^=nav]`匹配`<div id="nav-primary">`和`<div id="nav-secondary">`）。
- 为电子邮件链接指定样式（使用 `a[href^=mailto]`），请参阅 http://css-tricks.com/better-email-links-featuring-css-attribute-selectors。
- 为站外链接（使用 `a[href^=http]`和 `a[rel=external]`）、安全协议链接（使用 `a[href^=https]`）、特殊网站的链接（比如使用 `a[href*="paypal.com"]`）、新开窗口的链接（`a[target="_blank"]`）或回到本站首页的链接（使用 `a[href="http://myurl.com"]`或 `a[href="/index.html"]`）分别指定不同样式。

- 网站上线前进行空链接检查，请参阅 http://fuelyourcoding.com/unconventional-css3-link-checking。
- 为链接显示快捷键提示（使用 a:after{ content:'[' attr(accesskey) ']' }）。
- 为 blockquote 标签显示引用来源（使用 blockquote[cite]:after { content: ' - ' attr(cite) } ）。
- 根据 blockquote 标签的不同 cite 属性值而为其设置不同的样式。
- 把图片的替代文字(Alternative Text)作为标题显示(使用 img[alt]:after {content: attr(alt) })。
- 编写一个用户样式表单来隐藏网页广告，请参阅 http://24ways.org/2005/the-attribute-selector-for-fun-and-no-ad-profit。
- 为 IE 6 隐藏某些样式规则。

表4-3　属性选择器的浏览器支持度

IE	Firefox	Opera	Safaril	Chrome
支持，7+*	支持	支持	支持	支持

*IE 7及其之后版本支持所有的属性选择器，但有时候会出现问题。请仔细测试。

4.4.2　使用属性选择器按类型定位元素

既然网页上的照片和缩略图拥有众所周知的不同之处，我们便可以利用属性选择器来应对这些差异。在我们的例子中，差异在于地图的缩略图存放于名为“thumbnails”的目录，而照片则在“photos”目录下。目录名是 src 属性值的一部分，因此我们可以使用属性选择器根据 src 值的差异而分别对待不同的图片类型。

让我们先把地图缩略图从右浮动到左边：

```
img[src*=thumbnails] {
    float: left;
    margin: 0 20px 10px 0;
}
```

带*的属性选择器告诉浏览器去“找到所有 src 属性值包含‘thumbnails’字符串的元素”。这条规则会匹配到地图的缩略图：

```
<img src="thumbnails/map.png" width="100" height="100"
¬alt="">
```

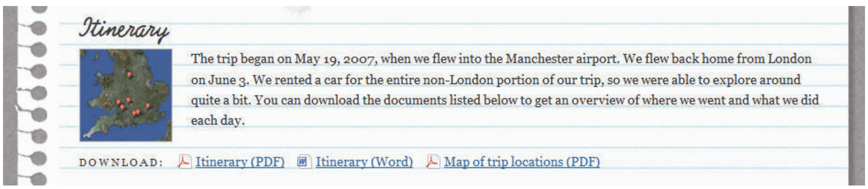


图 4-4　地图缩略图现在浮动到左侧了，而其他图片还是在右侧

说明　box-shadow 属性值里的-4px 是阴影的扩展半径，iOS 3 及之前版本上的 Safari 4 不支持这个特性，因此在这些浏览器上将没有阴影效果。你可以移除扩展半径的值来修正这个问题，但这样一来又会使阴影在 Safari 5 和 Chrome 上的效果打折。这里的权衡靠你自己。

现在让我们给照片增加一些样式让它们看起来像是宝丽来照片。请增加以下的新样式规则：

```
img[src*=photos] {
    padding: 5px 5px 30px 5px;
    background: #fff;
    -moz-box-shadow: 3px 6px 8px -4px #999;
    -webkit-box-shadow: 3px 6px 8px -4px #999;
    box-shadow: 3px 6px 8px -4px #999;
    -moz-transform: rotate(2deg);
    -o-transform: rotate(2deg);
    -webkit-transform: rotate(2deg);
    transform: rotate(2deg);
}
```

至此所有的照片四周都有了白色的边框，后面则有阴影效果，并旋转了一些角度（见图 4-5）。

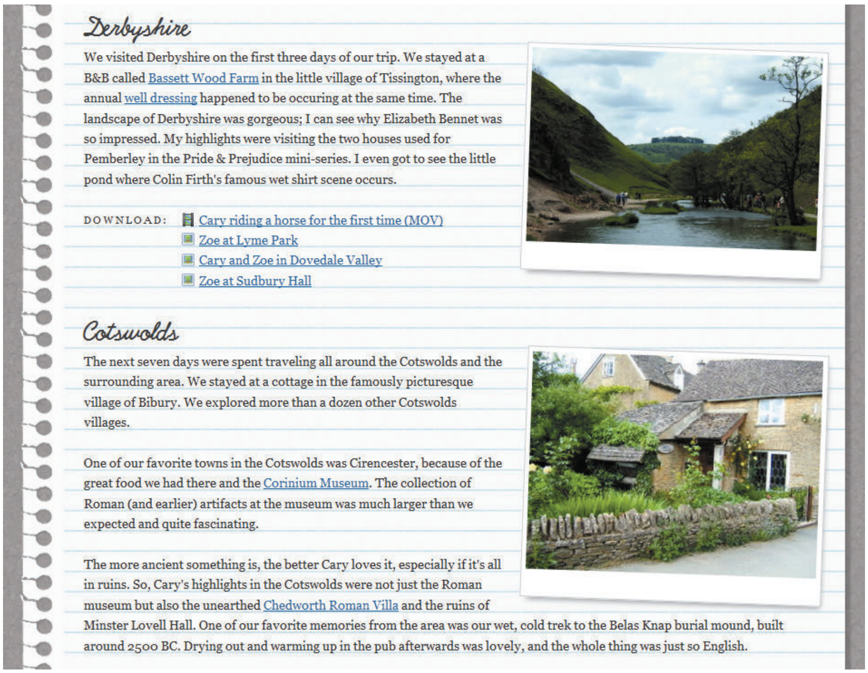


图 4-5　照片现在有了自己独特的效果——宝丽来风格

4.5　大功告成

至此我们对页面里的链接和图片完成了所有的样式定义。请在浏览器上检查我们的工作成果，

并对比图 4-6 和图 4-1。IE 7 及其之后的版本支持属性选择器，我们为 IE 6 加载了一个脚本来实现兼容，因此所作的大多数修改在 IE 上均是可见的。唯一不可见的是阴影和照片的旋转效果。

说明　完成版页面所展示的效果代码都在 selectors_final.html 页面里，它在你下载的本章示例文件中。



图 4-6　感谢 CSS3，它让我们完成的版页面拥有独特的样式（另见彩插图 4-6）

第5章 使用伪类来提高效率

前一章里，你已经学会了如何在不增加 ID 或类的情况下使用属性选择器来定位某个独立的链接和图片。而在本章中，你会学到更多的 CSS3 选择器，它们能帮助你保持代码干净和整洁，而且不依赖 JavaScript 或 Flash。我们将使用这些选择器为气泡对话框和文章页实现更多的视觉改进，更妙的是我们还会添加 CSS 的动画和变形来增强可用性。

你将学到

我们将为气泡对话框和文章页里的照片增加交替样式，还要为文章的当前章节建立一份目录，这会用到以下的 CSS3 特性。

- 使用`:nth-child()`伪类选取可替代元素。
- 使用`:nth-of-type()`伪类选取某个类型的可替代元素。
- 使用`:last-child`伪类为列表的最后一个元素定义不同的样式。
- 使用`:target`伪类来选取含有可扩展片段标识符（Fragment Identifier）的 URL 的元素。
- 使用变形（Transition）来平缓地更改某个属性的值。
- 使用动画（Animation）来掌控更复杂的视觉变化。

5.1 不使用 ID 和类来定位特定元素

正如属性选择器的功能一样，伪类和伪元素能在不指定 ID 和类的情况下用来选取特定的元素，还可以保持结构的整洁。伪类和伪元素可以将一段并不存在的 HTML 当作独立元素来定位，或是找到那些无法使用其他简单选择器就能定位到的切实存在的独特元素。比如，你可以使用`:first-line`伪元素对某个段落的首行进行格式化，尽管它并没有被某个 HTML 标签嵌套。这样看来，伪类和伪元素比属性选择器更为强大，因为它能让你定位那些从未拥有过 ID 或类的元素。

一般看来，伪类和伪元素对于 CSS3 来说并不是什么新鲜或特别的功能，但 CSS3 增加了若干个特别的伪类来帮助我们更精确地定位到文档的某个部分。它们之中的大部分都是结构伪类（Structural Pseudo-class）。

伪类和伪元素的差异何在

最简单的记忆方式是：伪类选取已设置过类的 HTML 元素，而伪元素选取的压根就不是 HTML 元素。

CSS 里的 4 个伪元素分别是`::first-line`、`::first-letter`、`::before`和`::after`。它们代表的是其他 HTML 元素的片段，而非该元素本身。它们也不属于文档树（Document Tree）的一部分，因此唯一可以定位到它们的方式便是使用伪元素选择器。

在 CSS3 的语法规范中，伪类以一个冒号打头，而伪元素是两个（它们曾经都使用一个冒号，而且至今有效）。每个选择器可以设置一个伪元素规则（比如`#article p::first-line`），但伪类则无此限制。

5.1.1　新的结构伪类

CSS3 引入了“结构伪类”的概念，它可以根据元素在文档树中的某些特性（比如相对位置）定位到它们。举例来说，`:first-child` 伪类指向那些父元素的首个子元素。这个元素是文档树中的一个独立的 HTML 元素，但它的特别之处在于它是第一个，而我们希望能在不增添类或 ID 的情况下就能够选取到它。

所有的结构伪类都是基于文档树的，它也称作文档对象模型（DOM），下面让我们快速地温习一下相关知识。*文档树*（Document Tree）是 HTML 页面的层级结构。它由元素、属性和文本组成，它们都是一个*节点*（Node）。文档树包含有多个层级是因为元素间互相嵌套（见图 5-1）。直接嵌入其他元素的元素被称作那些外部元素的*子元素*（Children）；而随着嵌套的继续深入，它们也成了*后代元素*（Descendants）。而那些外部元素被称为*父元素*（Parents）（一层之上）或*祖先元素*（Ancestors）（两层或以上）。位于相同嵌套层级的元素——换言之它们有共同的父节点——叫做*兄弟元素*（Siblings）。一个元素可以同时拥有以上部分甚至所有的称谓，正如你可以是某人的孩子，也同时是某人的父（母）亲，这些称谓都用来描述一个元素与另一个元素的关系所在。

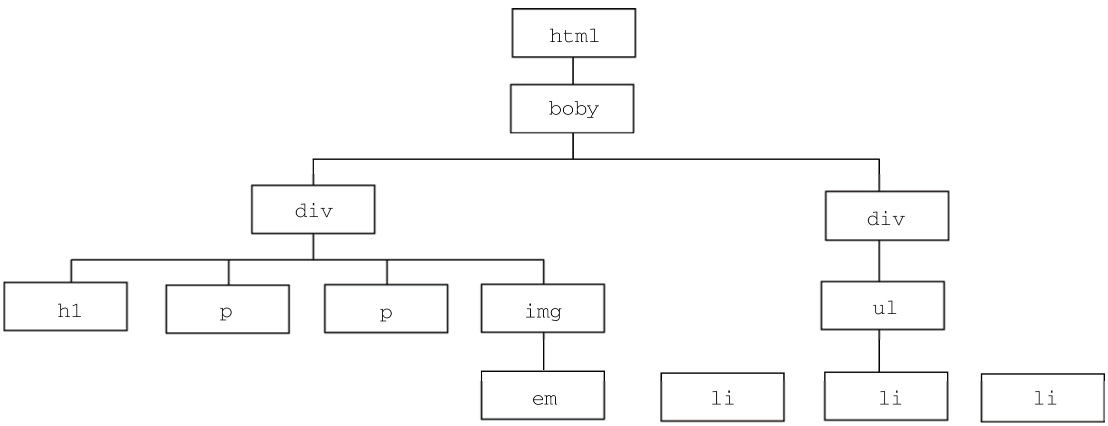


图 5-1　一个文档树示例，包含了祖先、后代、父、子和兄弟元素

我们已经弄清了所有的术语，现在可以看看有哪些可以建立元素间关系的方法了。表5-1罗列了全部结构伪类。

表5-1　结构伪类

伪　类	描　述
:root	选取文档的根元素。在HTML中，始终是指html元素
:nth-child()	根据其父元素的子元素的序号来选取元素
:nth-last-child()	与:nth-child()相似，但计算序号的方式不是从前往后，而是从后往前（倒数）
:nth-of-type()	根据其父元素的子元素的序号来选取元素，但只有符合给定类型（如p、img或其他）的元素才能参与排序
:nth-last-of-type()	与:nth-of-type()相似，但计算序号的方式不是从前往后，而是从后往前（倒数）
:first-child	选取某个父元素的第一个子元素（在图5-1中，h1元素便符合条件）
:last-child	选取某个父元素的最后一个子元素（在图5-1中，img元素便符合条件）
:first-of-type	选取某个父元素的第一个同类型兄弟元素（在图5-1中，第一个p元素可以通过p:first-of-type来选取）
:last-of-type	选取某个父元素的最后一个同类型兄弟元素
:only-child	选取某个父元素的唯一一个子元素（在图5-1中，ul元素便符合条件）
:only-of-type	选取某个父元素的唯一一个某类型的子元素
:empty	选取没有子元素或不包含文本的元素

除了:first-child伪类是属于CSS2.1之外，其他的结构伪类都是CSS3的新特性。它们为我们提供了精确定位到元素的新方式。

5.1.2　回归气泡对话框：颜色交替

我们可以用:nth-child()伪类使第2章评论页里的气泡对话框的背景色实现隔行交替。而且我们无需使用类或JavaScript。

1. **:nth-child()如何发挥作用**

最强大也最实用的结构伪类之一就是:nth-child()。我已经在上文中提到它可以根据其父元素的子元素的序号来选取元素，换句话说，它根据其之前有多少兄弟元素来选取元素。

请在选择器的括号里填入你想选取的元素的序号，比如li:nth-child(5)。这个选择器将匹配某个列表里的第5个li元素。括号里除了可以使用数字之外（选择器的参数），你还可以使用关键词odd（奇数）或even（偶数）来隔行选取元素，比如第2、4、6……个。但:nth-child()的强大还在于你可以将算式作为参数传入其中，依此创造更复杂的交替选择模式，甚至可以同时依序选取某个特定区域的子元素。算式的语法是an+b，其中a是需要你来设置的循环的周期，n是从0开始的计数器，而b是你想要设置的偏移值。以下是一个示例：

```
li:nth-child(3n+1)
```

n的值从0开始，并按周期循环递增，因此这个选择器的匹配情况如下：

$(3 \times 0) + 1 = 1$ = 第1个列表项

(3 × 1) + 1 = 4 = 第 4 个列表项

(3 × 2) + 1 = 7 = 第 7 个列表项

(3 × 3) + 1 = 10 =第 10 个列表项

以此类推！

尽管你也可以手动给第 1、4、7、10 个列表项加上类，但这样做不仅耗时耗力，而且容易被遗忘，还会增加页面的尺寸，但最重要的是——维护过程很痛苦。如果想在已经存在的项目中插入另外一个，你将不得不对插入处之后的项目全部重新指定类，因为它们的序号已经全部改变了。用`:nth-child()`伪类代替你跟踪序号的变化并自动匹配将会更为准确、高效。

千万不要因为数学计算而对`:nth-child()`产生抗拒。这里有一些不错的线上工具，你可以通过更改数值来即时地查看它是如何影响页面样式的，这将有助于更好地了解`:nth-child()`的作用。其中我最喜爱的一款工具是 http://leaverou.me/demos/nth.html，开发者是 Lea Verou，它不仅可以用来测试`:nth-child()`，还包括`:nth-last-child()`、`:nth-of-type()`和`:nth-last-of-type()`。

2. 斑马纹效果

`:nth-child()`最常见的应用之一便是使某个表格产生隔行变色的效果，一般也称作“斑马纹效果”（Zebra Striping）。它往往不仅是一种审美的提升，还能让你更轻松地浏览冗长的表格而不至于迷失目前的方位，并因此使可用性得到改善。

说明　这里有一份研究表明，斑马纹效果并没有你想象中那么实用——它的确能改善效果，只是效果并没有那么明显。需要了解更多信息请阅读 Jessica Enders 所撰的“Zebra Striping: More Data for the Case”（www.alistapart.com/articles/zebrastripingmoredataforthecase）。

没有`:nth-child()`的话，要使表格拥有斑马纹效果，必须隔行增加一个类，通常命名为`even`或`alt`，然后为这个类设置一个不同的背景色。必须手动增加这些类，或者用一段 JavaScript 来帮你实现。而这两种方式都不如`:nth-child()`来得高效。

你可以用`:nth-child()`算式来实现斑马纹，比如：算式`2n`将匹配所有的奇数行。而关键词`even`和`odd`则是更便于使用的一条捷径。我们将在评论页里通过关键词`even`来使气泡对话框产生隔行变色的效果。

开始之前，请到 www.stunningcss3.com 下载本章的示例文件并用编辑器打开 alternate_start.html 页面。其 CSS 的定义包含在页面`head`里的一个`style`元素里。该页面和第 2 章的文件相同，你也可以将第 2 章的最终页面作为本章的起始点。

现在，所有的气泡对话框都是相同的暗蓝色（见图 5-2）。改颜色的值为`hsla(182,44%,76%,.5)`。既然我们使用了 HSLA 格式，那么将交替色调整为一个稍显不同的颜色就相当简单了。你应该还记得色调的取值范围是 0 到 360，光谱从红色到紫色。因此如果你想要使交替色变得更绿一些，可以轻松地将色调值调小一些，比如用 160 来替换 182。而如果你想让交替色变得更蓝，请调高色调值，比如 200。



图 5-2　所有评论拥有相同的蓝绿色背景

让我们为交替色选用更蓝一些的颜色。我将使用 210 作为色调值来使其颜色变化显得更为明显。请在页面的 head 增加如下的 CSS 代码：

```
li:nth-child(even) blockquote {
   background-color: hsla(210,44%,76%,.5);
}
```

保存页面，然后用尚不落伍的浏览器查看效果。你将看到第 2 条和第 4 条评论变蓝了，而第 1 条和第 3 条仍保持蓝绿色（见图 5-3）。



图 5-3　偶数行的气泡对话框的背景色现在是烟蓝色

这个新的蓝色看起来与之前蓝绿色的饱和度并不相符，那么就让我们来提升它的饱和度使它变得更亮一些，再稍稍增加一点亮度值来衬托黑色文字：

```
li:nth-child(even) blockquote {
   background-color: hsla(210,70%,82%,.5);
}
```

现在交替色变成了更亮一点的暗蓝色（见图 5-4）。HSLA 帮我们更方便地选取一个互补色，而`:nth-child()`使我们更简单地实现了气泡对话框的颜色交替。尽管像日志评论这样的颜色交替并没有如表格那般真正提升了可用性，但你可以感受到使用`:nth-child()`按规则选取元素所带来的高效。

说明　完成版页面所展示的效果代码都在 alternate_final.html 页面里，它在你所下载的本章的示例文件中。



图 5-4　快速调整 HSLA 值使偶数行的对话气泡框变得更亮一些

表5-2　`:nth-child()`的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，3.5+	支持*	支持	支持

* Opera 支持`:nth-child()`，但当页面载入完成以后，用 JavaScript 生成的元素却不会更新样式。这个问题可通过增加一个`:last-child`声明来修复。欲获取更多信息，请参阅 http://www. quirksmode.org/css/nthchild.html。

`:nth-child()`伪类的来龙去脉

`:nth-child()`伪类是选择器（Selector）模块的一部分，你可以在 www.w3.org/tr/css3-selectors 找到它。它是一个结构伪类，可根据其父元素的子元素的序号来选取元素。

在`:nth-child()`的参数里，你可以填入一个数字（来选取特定序号的子元素）或关键词 odd 和 even（来隔行选取元素，即奇数或偶数序号的元素），还可以填入一个语法为 an+b 的算式（来选取特定组合的子元素）。在这个算式中，a 代表循环的周期，b 则代表偏移值。

a 和 b 均可以是负数值。而如果 a 赋值为 1，那么你可以忽略它（因为 1n+3 等于 n+3）。如果 b 赋值为 0，或者如果 a 和 b 的值相等，那么你可以忽略 b（因为 2n+0 和 2n+2 都等于 2n）。欲获取更多详细的信息，请参阅 http://reference.sitepoint.com/css/understandingnthchildexpressions。

除了斑马纹效果以外，你还可以将:nth-child()用在以下场景。

- 使一篇文章的头两段（或更多）呈现不同的样式（如果仅针对头两段，可使用-n+2）。
- 为 Top100 列表里的前 10 个项目使用较大的字号（使用-n+10）。
- 随着日志或微博的列表向下展现，使旧的条目呈现出较小的字号或更浅的颜色。
- 营造一种随意的界面（比如，每三个特性框便使用一种背景色，每四个则用另外一个，以此类推）。
- 每四个缩略图便强制换行或改变外补丁的值，以此来实现一个多行相册，请参阅 http://mondaybynoon.com/2010/03/18/css3-center-thumbnail-galleries。
- 为表格里特定的列定义不同的样式（比如，使包含数字的第三列靠右对齐）。
- 根据并排项目的数量改变宽度值以撑满剩余空间，请参阅http://andr3.net/blog/post/142。

3. 针对 IE 的兼容方案

IE 8 及之前的版本并不支持:nth-chid()，而 IE 9 已支持。在本例中颜色交替只是一个很小的视觉增强，IE 8 及之前版本的用户看不到也无妨。IE 将直接忽略我们新增的样式规则，并依旧的所有气泡对话框的原有颜色。看起来没有任何异常、不完整或丑陋之处。

要是你仍旧想为 IE 8 及之前的版本提供一个解决方案，需要用到 JavaScript。如果你要实现斑马纹效果，网上有大量的脚本可以帮助你隔行添加类。最适合你的脚本还是取决于项目本身，可以在 Google 里搜索“Zebra Stripe JavaScript”、“Zebra Stripe PHP”或其他更符合你需求的关键词。

除了上面的方式之外，你还可以使用一个脚本为 IE 增加高级选择器的支持，然后使用选择器实现你想要的效果，这当然包括但不限于斑马纹效果。其中一个脚本就是我们在上一章所使用过的由 Dean Edwards 开发的 IE 7.js（http://code.google.com/p/IE 7-js），但这一次你必须升级为 IE9.js 来获得对伪类的支持，IE 7.js 仅兼容了属性选择器和其他少数选择器。另一个不错的脚本也实现了对伪类的支持，它名叫 Selectivizr（http://selectivizr.com），不过如我们在第 4 章中提到的：要使它工作，还需要引入其他某个 JavaScript 库，比如 jQuery、MooTools 或 DOMAssistant。以上两个脚本都使 IE 能够识别业已存在的 CSS 并按照样式定义来渲染它们。

再次回顾第 4 章我们所提到的，很多 JavaScript 库内部已支持如:nth-child()这样的选择器，你可以将它写入自己的脚本并使其在 IE 上工作。但这个方法不会使目前已经定义的 CSS 代码生效，你将不得不在脚本中重新定义。请参看第 4 章的“使用 JavaScript 库”这一节以获取可供使用的脚本库链接。

5.1.3　回归照片：随机旋转

我们已经用:nth-child()改进了气泡对话框，接下来让我们回到第 3 章和第 4 章里我们已完成的文章页面，看看能为照片增加什么样的交替效果。现在，所有的照片都已旋转过，好让它

们看起来更真实。但由于旋转角度相同，它们看起来非常整齐（见图 5-5）。要是用:nth-child()来使不同的照片旋转不同角度并以此来增强界面的随意感和真实感应该非常不错。



图 5-5　所有的照片都微微向右旋转

然而，当你使用 img[src*=photos]:nth-child(even)选择器使所有偶数号的图片改向左旋转的时候，你可能会惊讶地发现最后两个图片却都向右转了，并没有实现交替。这是因为:nth-child()伪类仅选择拥有相同父元素的所有子元素。这些 img 元素与所有的 p 和 h2 是兄弟元素，因此全部这些元素都被:nth-child()选取到了。尽管选择器中包含了 img 元素，但那个选择器要求的是“找到所有 src 属性包含‘photos’字样的图片，然后对那些偶数号的子元素应用指定的样式”。如果你计算过 div 下 img、p 和 h2 元素的个数，会发现倒数第二张照片是 div 的第 29 个子元素，而最后一张照片位列第 37。因此，那条:nth-child()规则只选取偶数号的子元素，而不会对它俩应用样式，所以它们依旧向右旋转。

我们真正需要的选择器必须与:nth-child()相似，但它必须只计算特定类型的元素。幸运的是，CSS3 早已提供了这样的选择器——:nth-of-type()伪类。它的功能与:nth-child()相同，但仅将写在它前面的元素纳入计数。

让我们将它应用到页面里。请从本章的示例文件里找到并打开 rotate_start.html（或直接使用第 4 章的完成页面）。在页面 head 里增加如下新样式：

```
img[src*=photos]:nth-of-type(even) {
    -moz-transform: rotate(-2deg);
    -o-transform: rotate(-2deg);
    -webkit-transform: rotate(-2deg);
    transform: rotate(-2deg);
}
```

保存你的页面，然后在浏览器中查看是否每个偶数号的图片都改向左旋转了（见图 5-6）。这一次最后两张图没有倒向同一边，而是分别朝不同的方向旋转。

你可能注意到图 5-6 里第 1、3 和 5 号图片都向左旋转了，尽管选择器的要求是偶数号的图片才向左转。这是因为页面里所有照片之前还有另一个 img 元素——地图的缩略图。它的 img 标签使第 1、3 和 5 号照片成了总数上的第 2、4 和 6 号图片。:nth-of-type()伪类在计算元素个数时仅关心其类型——在本例中，元素类型指的是 img。整条选择器的意思是“找到所有 src 属性包含‘photos’字样的图片，然后对那些偶数号的 img 元素应用指定的样式”。

:nth-of-type()的威力

如果你想知道:nth-of-type()（或其他一些高级选择器）的真正威力，请访问 Harry Roberts 没用任何 ID 或类就实现的一个多列布局的示例页面（http://csswizardry.com/2010/04/building- sites-without-using-ids-or-classes）。所有的 div 都改用高级选择器来定位。它并未展现什么特效，只是一种页面实现的可能性和一份选择器的学习范例。

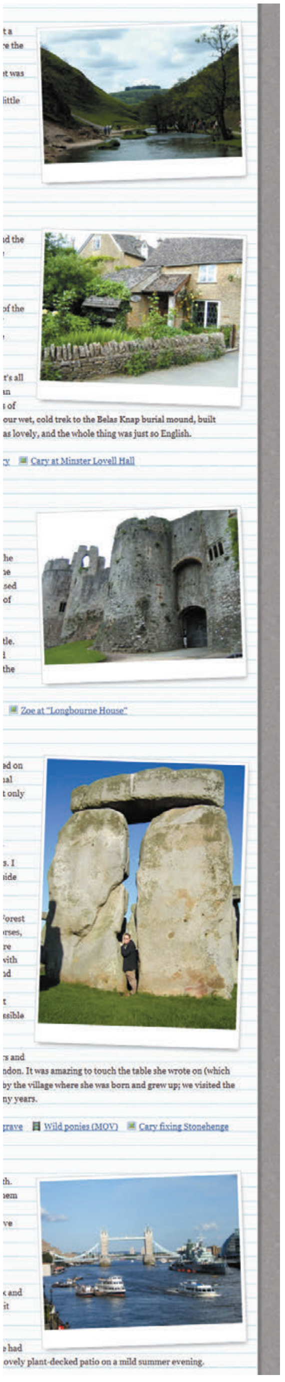


图 5-6　现在照片在左旋和右旋之间交替，看起来更真实了

:nth-of-type()伪类的来龙去脉

:nth-of-type()伪类是选择器（Selector）模块的一部分，你可以在 www.w3.org/tr/css3-selectors 找到它。它是一个结构伪类，可根据其父元素下同类型子元素的序号来选取元素。它接受的参数（填入括号）与:nth-child()一样。

除了旋转图片的需求以外，你还可以将:nth-of-type()用在以下场景。

- 营造一种有随意感的界面，效果不仅仅是实现旋转。
- 使文章中的图片交替着向左或向右浮动。
- 为一篇文章的头一段或几段指定不同的样式；如果有其他元素可能会使这些段落的序号变得靠后（比如有时 h2 或 img 会处于段落之前），这种情况下:nth-child()将不会如你所愿。
- 为一个定义列表的条目使用交替样式；由于每个 dt 可以对应一个或多个 dd 元素，那么你无法使用:nth-child()。
- 为文章中的 blockquote 元素实现交替样式。

说明　IE 对:nth-of-type()的支持情况与:nth-child()相同，而且 JavaScript 的兼容方案也一样。

说明　完成版页面所展示的效果代码都在 rotate_final.html 页面里，它在你所下载的本章的示例文件之中。

表5-3　:nth-of-type()的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持，3.5+	支持*	支持	支持

* Opera 对:nth-of-type()和:nth-child()的支持有相同的问题。

CSS3 中并无方法可以让浏览器计算只拥有特定属性的 img 元素。所有其他的 img 元素都和照片一起参与子元素的计数和排序。在本例的页面中，我们仅仅尝试使照片看起来随意一些，因此有其他图片扰乱了我们的设想其实并非坏事。:nth-of-type()伪类仍旧按计划发挥了作用，就算它并未依据我们的设想选取元素。

事实上，可以利用另一条:nth-of-type()规则来获得更随机的照片展现：

```
img[src*=photos]:nth-of-type(3n) {
    -moz-transform: rotate(1deg);
    -o-transform: rotate(1deg);
    -webkit-transform: rotate(1deg);
    transform: rotate(1deg);
}
```

这会使每三个图片的角度便向右调整一度。现在照片的旋转效果呈现出一副更随意的样子了：第 1 张负向旋转-2deg，第 2 张旋转 1deg，第 3 张负向旋转-2deg，第 4 张 2deg，而第 5 张则是 1deg（见图 5-7）。

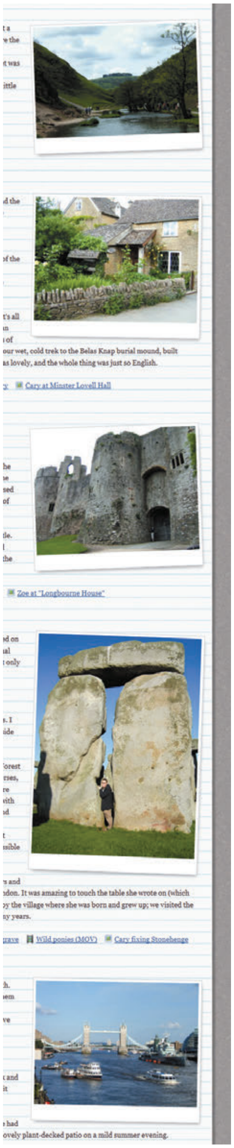


图 5-7　旋转变得更多样化了，看起来更随意、真实

即使是:nth-of-type()也可能无法满足你的所有要求，但它仍旧为你提供了不使用类和 ID 便能定位到元素的大量方法。

5.2　动态高亮显示页面的区域

你已经看到过两个例子，它们为你展示了如果使用 CSS3 的结构伪类在不使用类和 ID 还有 JavaScript 的情况下使页面的视觉效果得到提升。其他 CSS3 的伪类还能为你的页面增添更多的动态效果，比如当你点击页内链接后聚焦于页面下方某处时当前的区块会高亮呈现。这不仅仅是视觉效果的提升，还是对可用性的改善，因为它帮助浏览者找到他们在页面中的停留位置。

举例来讲，当你点击维基百科里某篇文章的引文编号时，页面会向下跳转到页面末尾与之相应的注释处。维基百科会高亮显示你所点击的那条注释，因此就不必在可能存在的成百上千个注释里找到与之对应的那一条（见图 5-8）。

当用户所点的条目位于页面底部而展现时却出现在浏览器窗口顶部时，上述方法特别管用。

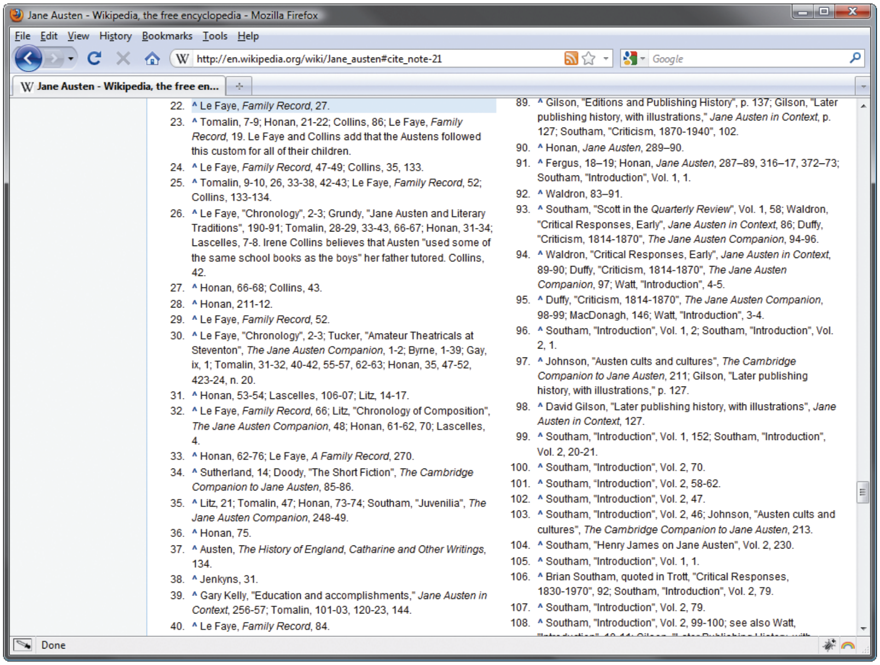


图 5-8　维基百科使用`:target` 伪类将所选脚注高亮显示为蓝色

你当然也可以使用 JavaScript 来高亮所选的脚注、标题或其他页面区域，但是用 CSS3 的`:target` 伪类将更为高效，不仅节省了开发时间，还缩短了页面的读取时间。

5.2.1　`:target`伪类

一些 URL 拥有片段标识符（Fragment Identifier），它由一个井号后跟一个锚点或元素 ID 组合而成，可以链接到页面的某个特定元素。图 5-8 中 http://en.wikipedia.org/wiki/Jane_austen#cite_note-21 这个地址就属于该类型。`:target` 伪类选取链接的目标元素，然后供你定义样式。

在维基百科上，当你点击脚注序号，与之对应的脚注的 li 元素便被选取。以下是维基百科为目标脚注定义的样式：

```
ol.references > li:target, sup.reference:target, span.citation:target {
    background-color: #DEF;
}
```

我们也可以用:target 伪类来为你的文章页面实现相似的效果。

5.2.2　增加目录

当前，文章页面并没有任何片段标识符来供我们链接。既然二级标题将文章自然地分割成章节，那么让我们先给它们增加 ID。

如果你完成了 rotate_start.html 页面，下面就可以继续修改它了，或者你也可以从本章的示例文件中打开 target_start.html，两个页面理应相同。在你的页面中，请为每个 h2 元素增加 id 属性。首先从“Derbyshire”开始，代码如下所示：

```
<h2 id="derbyshire">Derbyshire</h2>
<h2 id="cotswolds">Cotswolds</h2>
<h2 id="daytrips">Day-trips from the Cotswolds</h2>
<h2 id="winchester">Winchester and Surrounding Area</h2>
<h2 id="london">London</h2>
```

现在在页面的顶部增加目录链接到这些 h2 元素。请在“Itinerary”这个 h2 元素前增加如下列表：

```
<ul id="toc">
    <li><a href="#derbyshire">Derbyshire</a></li>
    <li><a href="#cotswolds">Cotswolds</a></li>
    <li><a href="#daytrips">Day Trips from the Cotswolds
    ¬</a></li>
    <li><a href="#winchester">Winchester and Surrounding
    ¬Area</a></li>
    <li><a href="#london">London</a></li>
</ul>
```

保存页面，然后在浏览器中浏览。目录中的链接应该会将你带到页面下方相应的二级标题处。我们还未给当前的二级标题增加任何特殊样式，但在做这一步之前，我们需要给目录加上样式。

说明　至此为止的所有修改都在 target_1.html 页面里，它在你下载的本章的示例文件之中。

---

1. 为目录定义样式

目录列表已经有一些非标准的样式了，因为页面里早已存在 ul 和 a 元素的样式规则，目录里的链接排列成一行，由空格将彼此分隔开来（见图 5-9）。让我们来进一步改进这个样式。

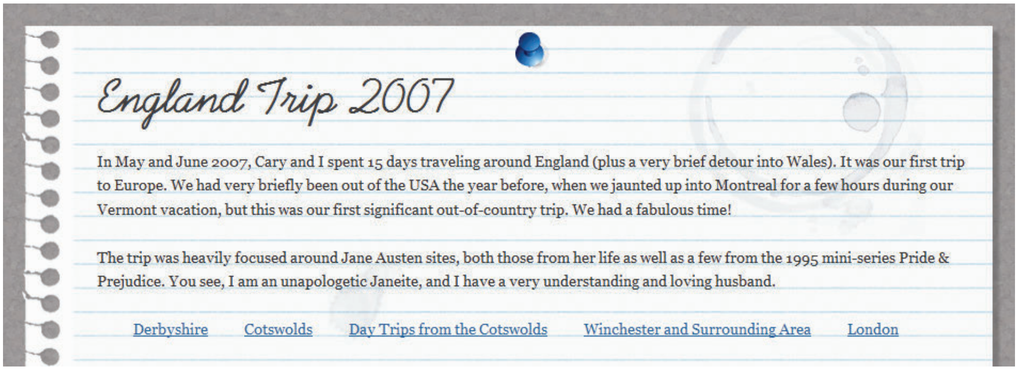


图 5-9　尚未定义任何特别样式之前的目录

先为列表增加一个手绘箭头的背景图：

```
#toc {
    background: url(images/arrow.gif) no-repeat top right;
    padding-top: 1.6em;
}
```

接下来，去掉 li 和 a 元素的左内边距值，改用右内边距，这样列表整体都向左对齐了：

```
#toc li {
    padding: 0 1.2em 0 0;
}
#toc a {
    padding-left: 0;
}
```

这里是使用另一个 CSS3 伪类的好机会。:last-child 伪类可以帮助你给某个父元素的最末一个子元素指定特别的样式。在这里，我们可以用它来去掉列表最后一个项目的右内边距：

```
#toc li:last-child {
    padding-right: 0;
}
```

这里所删除的右内边距会压缩最后一个项目所占据的空间，减少它在非必须的情形下折到第二行的几率。:last-child 伪类在移除某个列表、div 或表格列的最后一个项目的内边距、外补丁、边框或其他样式时显得非常实用。

图 5-10 展示了目前我们所定义的样式。列表看起来好多了，但还是显得太过单调。如果为每个列表项目增加一个小图标或数字，又将如何？



图 5-10　目录现在有了背景图，链接也有了更好的间距值

2. 用纯 CSS 实现数字“图标”

为了在列表项之前加上数字序号，我们可以用有序列表（ol 元素）来代替无序列表（ul 元素）。但是我们无法直接为浏览器增加的列表序号设置样式。当然还有一些方法可以变相地实现，但它们可以实现的效果有限，而且会在结构中增加垃圾代码。

另一套可选方案则是使用数字背景图。但这个方案也有它的缺点——会给页面增加 5 个 HTTP 请求。为了使负面影响最小化，你可以使用一项名为“CSS Sprites”（CSS 精灵）的技术，它能将所有图片合并为一张，然后通过调整背景图的位置来为每个列表项展示这张大图的一小部分从而实现我们的需求。但即使是用 CSS 精灵，你也仍旧需要多增加一次额外的 HTTP 请求，外加书写一些繁琐的 CSS 代码来应用这项技术。

除了使用图片之外，我们可以使用内容生成特性帮我们插入数字，就像在第 2 章里已经尝试过的那样。但这一次，我们将走得更远。我们不会把数字硬编码到 content 属性里——因为这样就需要为五个列表项书写五条不同的规则——而会使用一项 CSS 2.1 的特性，CSS 计数器来动态生成并自动实现数字的递增。

要使用计数器，必须首先使用 counter-reset 属性来命名并新建一套计数器：

```
#toc {
    background: url(images/line.png) no-repeat top right;
    padding-top: 1.6em;
    counter-reset: list;
}
```

> **关于 CSS 精灵技术的更多信息**
>
> 尽管 CSS3 的出现帮助我们省掉了许多之前用到 CSS 精灵的图片，但在很多情况下使用 CSS 精灵背景图片技术仍旧显得很方便。以下便是更多关于这项技术的信息。
>
> - “CSS Sprites: What They Are, Why They’re Cool, and How To Use Them”，作者 Chris Coyier（http://css-tricks.com/css-sprites）。
> - “CSS Sprites Workflow”，作者 Chris Coyier（http://css-tricks.com/css-sprites-workflow）。
> - “CSS Sprites: Useful Technique, or Potential Nuisance?”，作者 Louis Lazaris（www.smashingmagazine.com/2010/03/26/css-sprites-useful-technique-or-potential-nuisance）。

以上代码新建了一套计数器，姑且命名为“list”，现在你就可以把它应用到一系列元素上了（你也可以在 counter-reset 属性里为计数器设置一个起始值；默认由 0 开始，由于这正是我的需求，因此我没有在属性中设置起始值）。目录列表里的 li 元素序列便是我们将要应用“list”计数器的目标。要应用这个计数器，我们在#toc li 规则里使用 counter-increment 属性：

```
#toc li {
   padding: 0 1.2em 0 0;
   counter-increment: list;
}
```

这会告诉浏览器你想要对每个 li 元素进行递增计数，但它并不会将数字展示出来，这时你

需要用到 content 属性。现在用:before 伪类建立一条新规则以便在每个列表项的内容之前显示数字：

```
#toc li:before {
    content: counter(list);
}
```

说明 请记住，要使用计数器，你必须使用三个不同的属性：counter-reset、counter-increment 和 content。欲获取更多关于计数器的信息，请参阅由 David Storey 撰写的"Automatic Numbering with CSS Counters"一文（http://dev.opera.com/articles/view/automatic-numbering-with-css-counters）。

这条规则将告诉浏览器你想要显示的内容是一个名为"list"的计数器。使用它之后，数字序号就会神奇地出现在每个列表项之前了，从 1 开始，每条列表项递增 1 个数（见图 5-11）。



图 5-11 CSS 计数器在链接之前显示出递增的数字序号

我们可以把这些数字当作页面中的其他内容一样来设置样式。首先，让我们先使它们和文字发生浮动并在列表项之前增加一点儿左内边距，这样就能保持它们在同一直线：

```
#toc li:before {
   content: counter(list);
   float: left;
}
#toc a {
   float: left;
   padding-left: 5px;
}
```

接下来用 border-radius 属性给每个数字增加一个圆圈背景，使用和链接相同的暗蓝色，但半透明化：

```
#toc li:before {
    content: counter(list);
    float: left;
    width: 1.6em;
    height: 1.6em;
    -moz-border-radius: .8em;
    -webkit-border-radius: .8em;
    border-radius: .8em;
    background: #87B3CE;
    background: hsla(203,78%,36%,.5);
}
```

说明 IE 9 的确支持 border-radius 属性，但看样子（在此书撰写之时）并不适用于 CSS 生成的内容，因此在 IE 9 上圆圈将被方框所代替。

正如在第 2 章题为“使用 border-radius 来实现（椭）圆形”里所解释的那样：你可以将块元素的宽和高设置为相同的值（本例中是 1.6em），接着将 border-radius 属性的值设置为上述值的一半（即.8em），这样就能绘制出一个圆形。

从图 5-12 可以看到数字确实已经有了圆圈背景，但文字仍旧需要针对圆圈的位置做进一步的调整。请为#toc li:before 规则增加如下新定义：

```
color: #fff;
font-family: Arial, Helvetica, "Helvetica Neue",
sans-serif;
font-weight: bold;
text-decoration: none;
text-shadow: 0 1px 0 hsla(0,0%,0%,.6);
text-align: center;
```



图 5-12 使用 border-radius 属性来为数字增加圆圈背景

现在数字就绝对看起来像是真正的图片了（见图 5-13）。这样我们就在不依赖任何图片并不修改 HTML 结构的情况下实现了图标的界面制作。

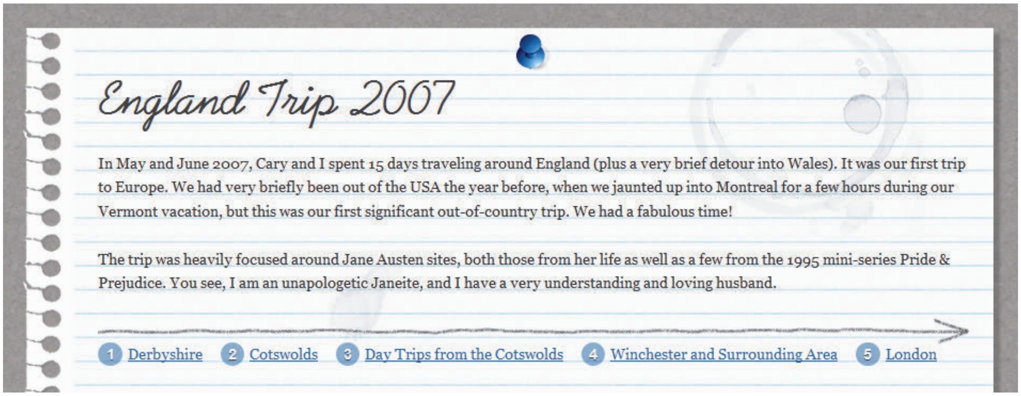
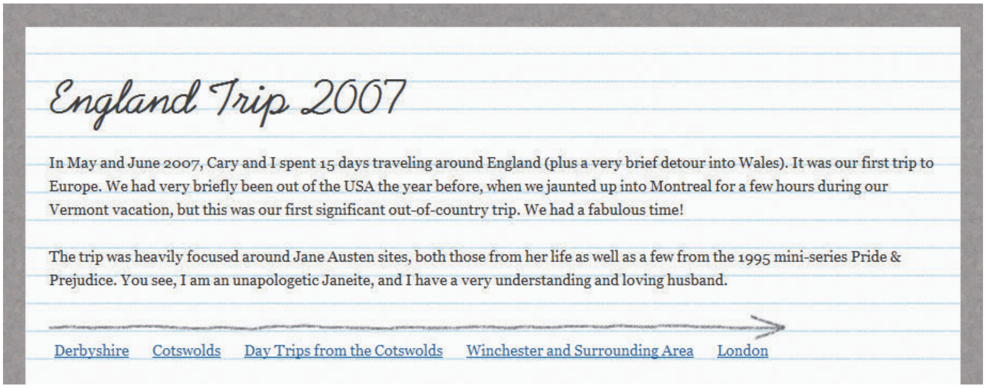


图 5-13 借助了 CSS3 的帮助，数字图片看起来就像是图片一样

不支持内容生成特性的浏览器将不会显示数字，那就不加理会吧（见图 5-14）。本例中，数字只是起装饰性作用，并非必要的内容，因此这也是一次可以接受的渐进增强实践。



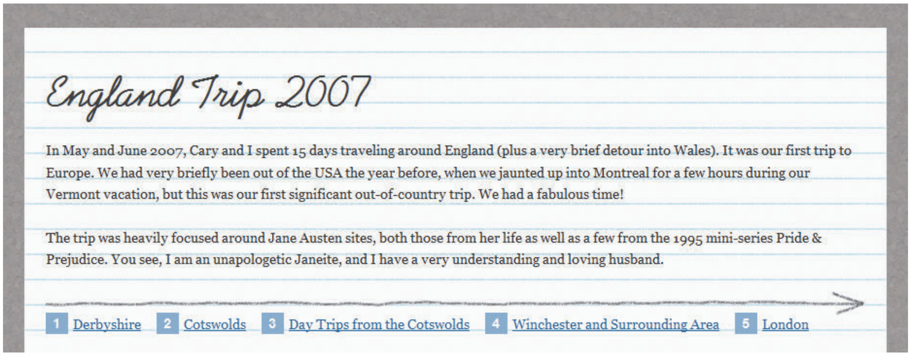


图 5-14　IE 6、IE 7（上）并不会显示数字，而 IE 8、IE 9（下）则会显示，但并无其他样式

说明　至此为止的所有修改都在 target_2.html 页面里，它在你下载的本章的示例文件之中。

5

5.2.3　更改聚焦区块的背景色

以上所有针对目录的样式设置还仅仅是准备工作，真正要做的现在才拉开帷幕：当你点击目录中的链接，页面会将聚焦区块进行高亮显示。当你点击链接时定位到的将是一个 h2 元素，因此我们所需的选择器是 h2:target。用这个选择器建立一条新规则，并将我们用作数字图标背景色的暗蓝色设置为它的背景色，但指定更高的半透明度：

```
h2:target {
   background-color: hsla(203,78%,36%,.2);
}
```

代码就是这般简洁。请保存页面，然后用浏览器打开，并点击其中一个链接。对应的标题将显示出一个半透明的淡蓝色背景（见图 5-15）。

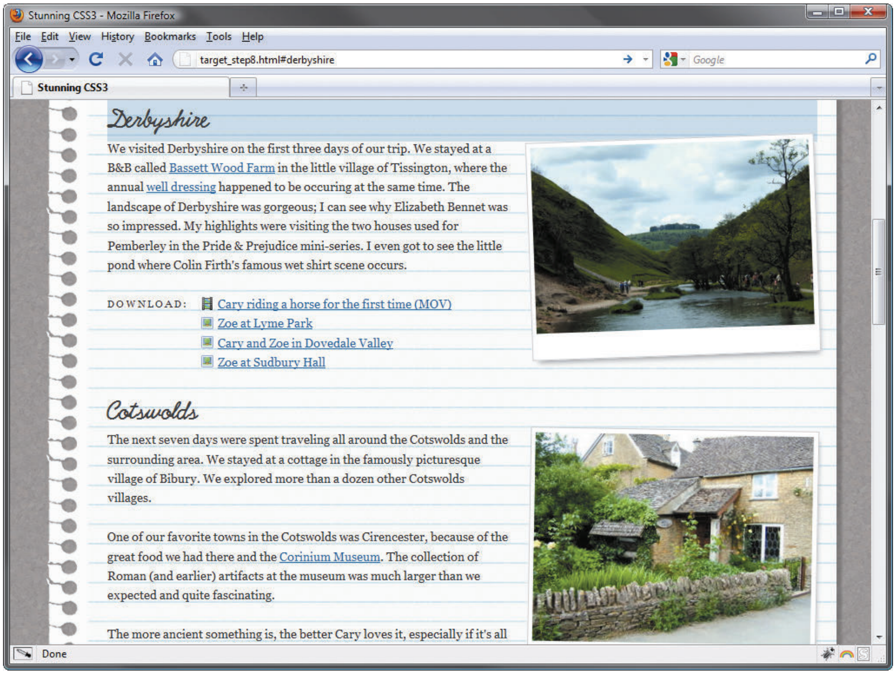


图 5-15　浏览器不仅将标题定位到窗口顶部，而且为其显示了背景

为了使界面变得更漂亮一点，你可以给文字增加一些左内边距和阴影效果：

```
h2:target {
   padding-left: 10px;
   background-color: hsla(203,78%,36%,.2);
   text-shadow: 1px 1px 2px #fff;
}
```

现在标题已经有了一个醒目但不突兀的高亮效果，当页面的焦点跳到下方时可以帮助用户定位（见图 5-16）。当你通过包含片段标识符的 URL 进入页面时同样会显示高亮效果。

说明　至此为止的所有修改都在 target_3.html 页面里，它在你下载的本章的示例文件之中。



图 5-16　高亮标题现在有了轻微的阴影，左内边距可使它远离背景的左边界

对整个区块进行高亮显示

在我们的示例页面里，如果将高亮标题改为高亮整个文章区块就显得有些过火，但这个方案在其他情况下也许会很有效。为了实现这个需求，你需要给目标区块包裹一些元素，比如 `div` 或 `section` 元素（顺理成章）。给这些包裹元素赋予一个 ID，并为其加上`:target` 伪类来实现定位。

一个可选的替代方案是将一个相邻兄弟选择器和`:target` 伪类结合起来使用，比如：`dt:target + dd`。然而，这个方案仅当你知晓聚焦之后有多少元素以及它们是什么类型的情况下有效。比如说，上一句给出的示例选择器就只匹配到目标 `dt` 之后的一个 `dd` 元素，将不会匹配 `dt` 元素之后可能存在的更多 `dd` 元素。

`:target` 伪类的来龙去脉

`:target` 伪类是选择器（Selector）模块的一部分，你可以在 www.w3.org/tr/css3-selectors 找到它。它帮助你选取一个元素，该元素是一个包含片段标识符的 URL 的引用。

除了用来将当前页面区块的标题高亮显示之外，你还可以将`:target` 用在以下场景。

- 高亮显示脚注。
- 在一个目标标题旁显示解释性文字，这样用户就能了解更多关于目前所处位置的上下文信息，请参阅 http://web-graphics.com/mtarchive/001454.php。
- 从一堆相互层叠的盒容器或图片中突出显示其中一个项目，请参阅 http://virtuelvis. com/archives/2003/07/target-fun。
- 标签化的内容块，请参阅http://css-tricks.com/css3-tabs。
- “手风琴式”（Accordion）目录或可折叠的内容块，请参阅 www.paulrhayes.com/2009-06/accordion-using-only-css 和www.thecssninja.com/css/accordian-effect-using-css。
- 幻灯片效果，请参阅 www.dinnermint.org/css/using-css3s-target-pseudo-class-to-make-a-lideshow 和www.nealgrosskopf.com/tech/thread.php?pid=45。
- 相册，请参阅 www.tobypitman.com/pure-css-sliding-image-gallery、www.IE 7nomore.com/fun/scroll 和www.IE 7nomore.com/fun/slideshow。
- 模态窗或“灯箱”（Lightbox）效果，请参阅 http://sixrevisions.com/css/semantic-css3-lightboxes 和 www.thecssninja.com/css/futurebox2。

请注意其中几项技巧或许用 JavaScript 来控制效果会更好，因为纯 CSS 的版本可能存在潜在的易用性和可用性问题。也就是说，它们也许在某些特定的环境下会很实用，也能够为你提供一些运用`:target` 的新想法，所以我将它们罗列在这里来激发你的灵感。

表5-4 `:target`伪类的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
支持，9+	支持	部分支持*	支持	支持

* Opera 支持`:target` 伪类，但当你使用前进和后退按钮跳离目标元素时，`:target` 样式却不会被移除。

针对 IE 的兼容方案

IE 8 及之前的版本并不支持:`target` 伪类，但 IE 用户点击目录里的链接时仍将跳转到相应的标题，但很显然标题不会高亮显示。既然页内链接的正常功能并无异常，IE 8 及之前版本的用户恐怕并不会怀疑"这里缺少了些什么功能"。用户恐怕也不会因为没有看到高亮效果就迷失于我们有限的页面内容。

然而，在包含更多内容的真实页面里，高亮效果就可能成为一项确保页面可用性的重要特性。可以想象一个拥有成百上千个脚注的维基百科页面——高亮效果就真的非常实用了。再比如作为跳转目标的区块离页面底部很近，浏览器无法将它显示在窗口可视范围的顶部——这同样会令用户感到迷惑。因此，如果你的页面必须要兼容 IE，你就需要用到 JavaScript。这里有一些脚本可供选择。

- "Suckerfish :target"（http://www.htmldog.com/articles/suckerfish/target），作者是 Patrick Griffiths 和 Dan Webb。
- "Improving the usability of within-page links "（http://dev.opera.com/articles/view/improving-the-usability-of-within-page-l），作者是 Bruce Lawson。
- "Fragment Highlight"（http://dorward.me.uk/software/frag），作者是 David Dorward。

5.2.4　用纯CSS实现动画

另一项很绚丽的改进是给标题的背景色加上淡入淡出效果，不断变化的色彩可能会直接引起浏览者的注意，甚至比突然发生的变化更有效。

你可以用 JavaScript 来实现上述效果。运用了该效果的一个流行案例被称作"黄色背景的淡出技术"（Yellow Fade Technique）。这项技术由 37signals 在其著名的 Web 应用——"Basecamp"里开创并命名。当你做了一次更改，被改动的部分将会高亮显示一个黄色背景且持续一段时间，接着黄色背景会慢慢消失 。它会让用户更多地注意到变更的部分，从而提升了可用性——帮助用户找到自己所在的位置并注意到页面里最重要的信息。可以阅读 37signals 的日志（http://37signals.com/svn/archives/000558.php）来了解该团队对"黄色背景的淡出技术"的解释。

当然，我们也可以抛开 JavaScript 并改用 CSS3 来实现相同的效果。基于 Webkit 内核的浏览器、Opera 和 Firefox 4 都支持过渡（Transition）特性，而基于 Webkit 内核的浏览器还支持用纯 CSS 的实现的动画（Animation）。过渡特性可以用来实现聚焦到标题所出现的颜色淡入和跳离时的淡出效果。动画特性可以用来实现聚焦到标题时出现的颜色淡入或淡出——或两者间的连续演化。我们将分别试试这两种不同的方案。

1."请稍等，这令我很不舒服"

在继续开发之前，先让我们暂停一下来缓和一下你潜在的焦虑。我知道 CSS 过渡和动画的实现对某些人来说并不是那么容易接受。

首先，它们的浏览器支持度并不好。就在本书撰写之时，过渡特性仅兼容基于 Webkit 内核的浏览器、Opera 10.5 及其最新版和 Firefox 4——它们仅占很小的市场份额。动画特性更是只兼容基于 Webkit 内核的浏览器。正因为如此，我想比起 CSS3 的其他特性你必须更谨慎地使用它们。

但只要你确认需要实现的效果是真的用来点缀页面的，那么较低的浏览器支持度就不该成为远离它们的理由。这样的话，运用它们不会伤害到任何人，而仅仅会多占用你一点点的时间和精力。需要补充的是，一旦支持度得到提升，你的页面——还有你的 CSS 技巧——将成为个中翘楚。

另一个被很多人所关注的问题是过渡和动画二者（尤其是动画）都涉足了“行为”（Behavior）领域，而不纯粹是“样式”（Style）。因此，（有些人认为）这些效果不应由 CSS 来实现，它们应该是 JavaScript、其他脚本、程序或 Flash 的活儿。

我在某种程度上赞同这个观点。动画是一个常见的行为，但它也同样是一个常见的样式。请想象一个正不断发光的按钮。“发光”对于按钮来说是一项行为呢？还是一个简单的视觉效果——样式呢？Jimmy Cuadra 在其“CSS3 过渡和动画：展现还是行为？”（CSS3 transitions and animation: Presentation or behavior?）（www.jimmycuadra. com/blog/12-css3-transitions-and-animation-presentation-or-behavior）一文中将这类效果称作“展现型行为”（Presentational Behavior）。我喜欢他对展现和行为所作的区别描述：

我们一般认为“展现”是事物看起来是什么，“行为”是事物做了什么。我认为“展现”是任何不会从根本上改变页面的东西，而“行为”则是任何会影响文档的结构或数据，或对显性的用户交互有帮助的事物。

Elliot Swan 则在一则评论（http://mondaybynoon.com/2009/05/04/covering-the-implication-and-basics-of-css-animation/#comment-9099）中对过渡和动画给出了另一种诠释：

> 我并不把过渡和动画看成是“样式”或“行为”中的任何一种，它们应该属于“效果”的范畴。（或者你也可以争辩说“效果”其实是“样式”和“行为”综合作用的结果。）

这种“展现式行为”（Presentational Behavior）或“效果”并不是从 CSS3 开始才有。早在 CSS 2.1 就有包含“行为”意味的样式出现，即`:hover`、`:focus` 和`:active` 伪类。当你悬停于某个按钮时它的颜色发生了改变，这是一个“行为”，但该“行为”是一种装饰以及对可用性的增强，而对内容或功能来说它并不是必须的。CSS3 只是对它进行了扩展，让你使用 CSS 就能更广泛地掌控动态的样式效果。我认为通过样式语言来控制简单的装饰性动画是行得通且可接受的，而更复杂或包含更多行为的动画则应该保留在脚本语言或 Flash 领域。

诚然，CSS 动画可能被滥用。它不该被用来实现重要的行为，异常复杂的动画用 Flash 来实现会显得更优雅和高效，但有一些人还是会选择用 CSS 来实现。对这种行为我只能表示遗憾，但这就是 CSS 的命运，对于任何 CSS 技术来说都一样。这些恶魔般的 Web 设计师总是能将纯洁的 CSS 捻碎在他们同样恶魔般的网站里。

不用关心这些“恶魔设计师”会用 CSS 动画和过渡来做什么。你只需保证自己带着责任感去有效地运用它们。毕竟它们有其自身的诸多好处。

2. CSS 过渡和动画的益处

CSS 过渡和动画最大的优点之一便是——一旦你了解它们的语法，那么初期部署和后期维护都将比同样效果的 JavaScript 和 Flash 代码要简单得多（不过请确保你恰当地使用了 CSS3 动画——较复杂的动画使用 Flash 完成仍更简单，无论如何你不该用 CSS3 动画来做复杂的事情）。

而且 CSS 是免费的，但使用 Flash 来制作动画则绝对不是。

而提到对用户的好处，CSS 过渡和动画并不依赖 JavaScript 的启用状态或 Flash 插件安装与否，它的效果由浏览器内置的功能生成。某些用户会禁用 JavaScript，而 Flash 也看起来永远不会出现在 iPhone、iPod Touch 和 iPad 的 iOS 上了。所以尽管目前浏览器对 CSS 动画的支持度还不高，但在未来，当支持度得以提升之后，它可能就成了你将简单的装饰性效果展示给最广泛受众的最佳方式了。

CSS3 过渡和动画还有效率上的优点。你无需使用任何外部 JS 或 SWF 文件来执行它，HTTP 请求也会因此减少。有时它们对用户机器的资源耗费会更少，至少和 JavaScript 的版本相比确是如此。但这很大程度上取决于动画的实现和用来对比的技术本身——Flash 版本对处理器的资源耗费通常比同等效果的 CSS3 版本要低。因此我再次重申，请务必只用过渡和动画来实现相对简单的效果，而且需要经过细致的测试。

说明　请参考 Flash 和 CSS3 两个版本的广告实现对比：www.sencha.com/blog/2010/07/20/html5-family-css3-ads-versus-flash-ads。

我并不是说 CSS 过渡和动画就是一粒万灵药或是它们并无缺点。只要保证在合适的场景下小心地使用，它们就会是非常实用的工具。下面就让我们看看如何实践这一点。

3. 使用 CSS 过渡实现色彩淡入

为当前标题的背景色实现淡入效果的首选方案便是使用过渡。这其实就是最简单的 CSS 动画了。你可以用过渡来简单地实现两个不同样式之间平滑地渐进式变化，而不会出现当你悬停、聚焦某元素或其他状态发生改变时样式突然地跳跃性变化。

要设置一次过渡，你需要告诉浏览器哪个属性你需要对其进行渐进式变化（使用 `transition-property`）和该变化会持续多少秒（使用 `transition-duration`）。你还可以选择性地增加一个过渡开始时的延迟值（`transition-delay`）和过渡期间速度的变化节奏（`transition-timing-function`）。

说明　你可以通过以下这个实用的示例来查看每种过渡节奏函数：http://css3.bradshawenterprises. com。

所有这些属性可以合并为一个简写属性：`transition`。把它加入 `h2` 规则，还包括三个浏览器的私有属性：

```
h2 {
    clear: left;
    margin: 0 0 -.14em 0;
    color: #414141;
    font-family: Prelude, Helvetica, "Helvetica Neue",
                 Arial, sans-serif;
```

```
    font-size: 2.17em;
    font-weight: bold;
    -moz-transition: background-color 1s ease-out;
    -o-transition: background-color 1s ease-out;
    -webkit-transition: background-color 1s ease-out;
    transition: background-color 1s ease-out;
}
```

在这里，我们已告诉浏览器：只要某个 h2 元素的 background-color 属性值发生变化，我们就希望它能渐进式地持续 1 秒钟。我们也将 transition-timing-function 属性指定为 ease-out 以期动画能在收尾时减慢速度。

过渡效果很难用静态的书籍描绘出来，但图 5-17 大致显示出当页面定位到标题时其背景色是如何渐进式变化的。当你按下浏览器的后退键来跳离标题时，过渡将反向播放，即从蓝色变为透明。

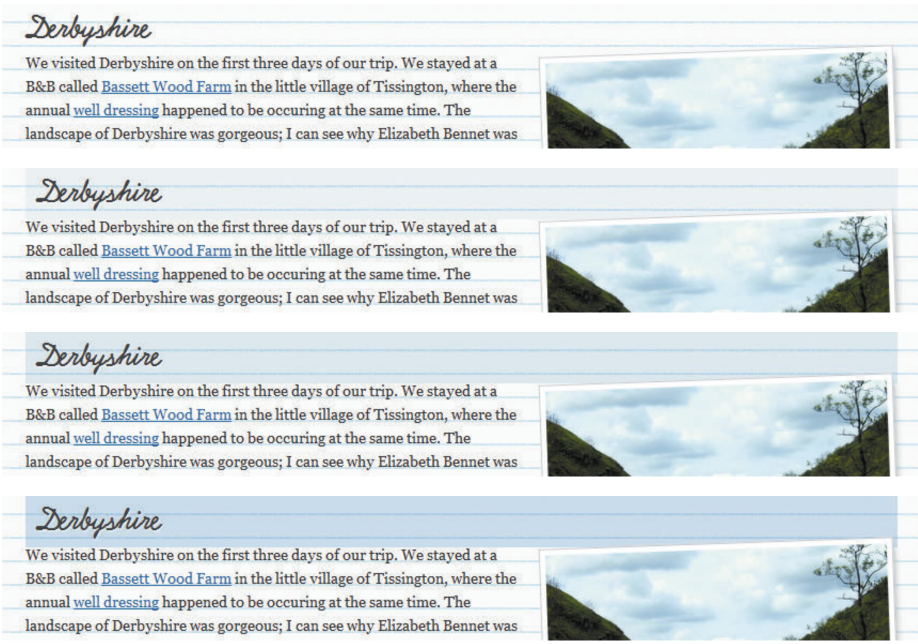


图 5-17　过渡持续了 1 秒钟以后，标题的背景色从透明逐渐变成了蓝色

哪些属性可以用来产生过渡效果

并非所有的属性都能用来产生过渡效果。W3C 把那些可以用作过渡的称作“可产生动画的属性”（Animatable Properties），并把它们罗列在 www.w3.org/tr/css3-transitions/#animatable-properties。这也是为何我在 h2:target 的规则里使用 background-color 而不用 background 简写属性的缘故，background-color 可以用来产生过渡效果，但 background

不行。或者应该说，它不该产生转换。Webkit 和 Firefox 并不遵守该规则，但 Opera 却不然，所以如果我们用 background 属性替换了 background-color，那么过渡效果就不会正常工作在 Opera 上。

我们也可以将过渡的相关代码从 h2 的规则里取出来放到 h2:target 里。但如果我们真的这样做了，过渡效果就只会当页面聚焦于标题的时候才会出现；而当焦点离开标题时，过渡效果便不会逆向播放了，而是突然地变回透明背景。而且，目前 Opera 只支持元素原始状态时的过渡，因此如果将过渡写到 h2:target 规则里，Opera 将无法看到效果。这看起来是一个错误的行为，但尚在制定中的 W3C 规范并未将这一点描述清楚。

说明　想要体验过渡的效果，请用支持过渡特性的浏览器打开 target_4.html 文件，该文件包含在本章的示例文件里，它包含了至此为止的所有修改。

除了背景色变化之外，我们还可以把左内边距加入到高亮标题的过渡代码里，这会创造出文字向右滑动的效果。你可以直接将内边距的过渡代码写到相同的 transition 属性里，用逗号隔开：

```
h2 {
    clear: left;
    margin: 0 0 -.14em 0;
    color: #414141;
    font-family: Prelude, Helvetica, "Helvetica Neue",
                 Arial, sans-serif;
    font-size: 2.17em;
    font-weight: bold;
    -moz-transition: background-color 1s ease-out,
                     padding-left .5s ease-out;
    -o-transition: background-color 1s ease-out,
                   padding-left .5s ease-out;
    -webkit-transition: background-color 1s ease-out,
                        padding-left .5s ease-out;
    transition: background-color 1s ease-out,
                padding-left .5s ease-out;
}
```

transition 属性的来龙去脉

transition 属性是过渡（Transition）模块的一部分，你可以在 www.w3.org/tr/css3-transitions 找到它。它是 transition-property、transition-duration、transition-timing-function 和 transition-delay 的简写形式。利用它，你可以使元素在状态变化时将某个属性值逐渐地变化成另一个值。

要实现一次过渡效果，必须设置的值是 transition-property（指定该元素需要进行渐进式变化的属性）和 transition-duration（指定变化需要持续的时间——如果留空，则

默认为 0)。其他参数是可选的。

一个元素的多个属性可以同时进行过渡，你需要将它们写在同一个 `transition` 属性里，并用逗号隔开。你也可以指定 `transition-property` 的值为 `all` 使该元素的所有属性都发生过渡。

但并不是每个属性都可以发生转换；请浏览 www.w3.org/tr/css3-transitions/#animatable-properties-来获取支持过渡的属性的完整列表。所有支持过渡特性的浏览器都支持这些属性中的大多数，但也存在诸多的例外恕我不能全部罗列于此。

欲获取更多有关过渡的语法细节和示例，请参阅 www.webdesignerdepot.com/2010/01/css-transitions-101 和 http://thinkvitamin.com/design/sexy-interactions-with-css-transitions。

除了用来实现背景色的淡入效果之外，你还可以将 `transition` 用在以下场景。

- 逐渐地改变按钮、标签或链接的悬停/聚焦和非悬停/失焦状态。
- 当鼠标悬停于图片时逐渐变亮（对不透明度进行过渡）。
- 使相互层叠在一起的图片实现交替渐变（比如用彩色照替换黑白照，或事物状态的前后对比图），请参阅 http://trentwalton.com/2010/03/30/css3-transition-delay 和 http://css3.bradshawenterprises.com。
- 当悬停时使图标缓慢变色（通过透明区域露出背景，之后再转换背景色），请参阅 www.ackernant.com/twtorial/rockingicon-with-css。
- 渐渐地显示出工具提示(tooltip)和信息块，请参阅www.zurb.com/playground/drop-in-modals。
- 实现物体发光或收缩的效果（使其宽度、高度产生过渡，或变形其缩放值），请参阅 www.zurb.com/playground/css3-polaroids 和 www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html。
- 使元素滑入界面之内（使其宽度、高度或位置产生过渡或变形），可用于相册、抽屉式目录或特色内容的幻灯片等，请参阅 www.nealgrosskopf.com/tech/thread.php?pid=45、http://dev.opera.com/articles/view/css3-show-and-hide、www.impressivewebs.com/animated-sprites-css3 和 http://css3.bradshawenterprises.com。
- 创造一副移动的背景图，请参阅 www.paulrhayes.com/2009-04/auto-scrolling-parallax-effect-without-javascript。

表5-5　**transition**属性的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	支持，需要-moz-前缀，4+	支持，需要-o-前缀，10.5+	支持，需要-webkit-前缀	支持，需要-webkit-前缀

至此，内边距将随着背景色的变化逐渐从 `0` 变到 `10px`（见图 5-18）。这两个过渡运行起来，平滑与否取决于你的浏览器（Webkit 似乎表现得最平滑，而 Firefox 则忽快忽慢），但这些简单的效果在所有支持该特性的浏览器上表现得都很好。

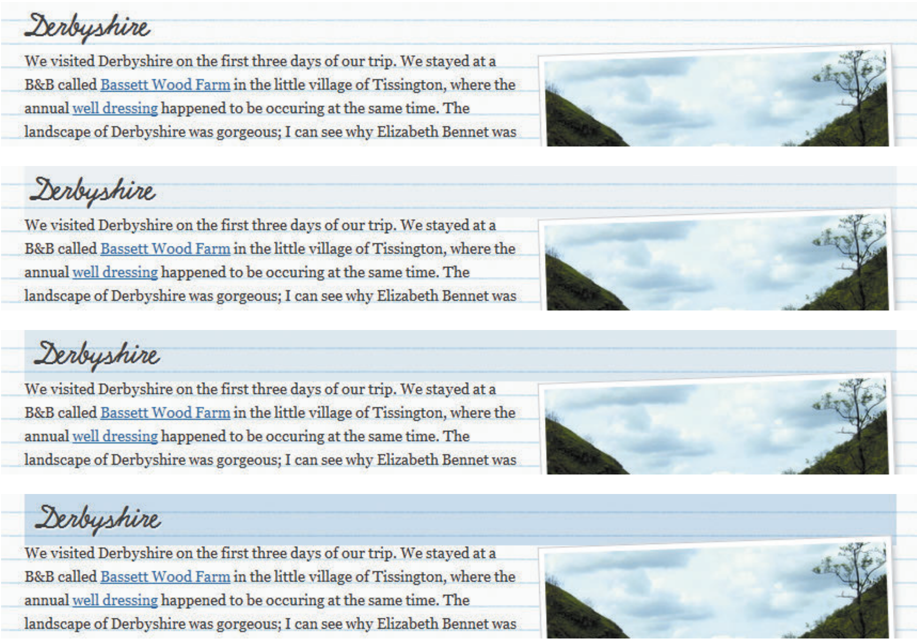


图 5-18　过渡完成之后，标题背景变暗而文字向右移动了 10 像素

4. 使用 CSS 动画使颜色淡出

我认为短暂地显示出当前标题的背景色然后再淡出，会比淡入效果更有效和醒目。由于过渡比动画拥有更广的浏览器支持度（请记住目前只有 Webkit 支持动画特性），如果能用过渡来实现这一效果将更好。但遗憾的是，过渡无法实现该需求，因为我们要求标题的背景色（聚焦之前）从透明变成蓝色（聚焦之后），接着再变回透明（聚焦一秒之后）。这个效果有三个变化点，而过渡仅能完成两个值之间的变化。

而 CSS3 动画理论上支持无限次变化。你可以设置一系列的关键帧（KeyFrame），即动画过程中变化点，每帧都有它自己的样式。接着，浏览器依照顺序平滑地在关键帧中过渡，慢慢地变化所有设置过的属性。

欲创建一个 CSS 动画，你首先需要为其指定一个名称，并定义每一个关键帧里所要发生的变化。你必须使用`@keyframes`创建规则，但既然目前只有 Webkit 支持动画特性，我们就需要书写两条规则：一条有`-webkit-`浏览器前缀，一条没有——这是为了向后兼容。请将下面的规则加入你页面的 CSS 代码里：

```
@-webkit-keyframes fade {
    0% {background: hsla(203,78%,36%,.2);}
    100% {background: none;}
}
@keyframes fade {
    0% {background: hsla(203,78%,36%,.2);}
    100% {background: none;}
}
```

我们将动画命名为“fade”，然后为其设定了两个关键帧：一帧是整个过程 0%（换而言之，即起始点），而另一帧是 100%（即结束点）。我们也可以使用关键词 from 和 to 来代替 0%和 100%。

目前已经设定好了想要的动画内容，下面需要将它赋予某个元素。我们希望动画在 h2 元素上产生，因此就在已有的 h2:target 规则下建立一个新的规则（很快将会发现为什么在这里没把代码写入已存在的规则里）：

```
h2:target{
    padding: 0;
    background: none;
    -webkit-animation-name: fade;
    -webkit-animation-duration: 2s;
    -webkit-animation-iteration-count: 1;
    -webkit-animation-timing-function: ease-in;
}
```

上述代码告诉浏览器该元素的动画名叫“fade”（使用-webkit-animation-name 属性），我们希望该动画持续两秒（使用-webkit-animation-duration 属性），并且仅执行一次（使用-webkit-animation-iteration-count 属性）。我们还告诉浏览器我们希望动画使用“缓入”（Ease in）效果，即动画的开始比结束时稍慢。

你也可以将所有这些属性合并到 animation 简写属性。下面我们将拥有-webkit-前缀的相关属性简写成-webkit-animation 属性，同样也增加一个无前缀的 animation 属性给 h2:target 规则：

```
h2:target{
    padding: 0;
    background: none;
    -webkit-animation: fade 2s ease-in 1;
    animation: fade 2s ease-in 1;
}
```

这条新建的 h2:target 规则还删除了前一条规则里定义的左内边距和背景色。如果我们没有删除背景色，动画将会执行一次，接着当它结束时，会突然跳回蓝色并一直保持住。我们需要透明的标题背景色，以便动画能完全地控制背景色，因此当动画结束时，我们希望标题的背景色能一直保持为透明状态。

删除内边距这一设置，从另一方面讲，并不是必需的。我选择删除它是因为一旦背景色消失，文字的缩进就没了意义。为了解决这个问题，我们可以为内边距增加动画——使它从 10px 缩减至 0，即我们在之前的过渡里所设置的相反过程。但我发现在本例中，内边距的变化反而会令人分心，它看起来更适合淡入效果，而不是淡出。我决定简化效果，去掉了内边距。

保存页面，然后在 Safari 或 Chrome 中浏览它。当你点击一个链接时，与之对应的标题背景将立即变为蓝色，并缓慢而平滑地消失（这个过程很像图 5-17，但顺序要倒转过来）。

假如蓝色在某个时刻停留一会儿然后才消失，效果应该会更妙。animation-delay 属性可以帮助你延迟动画的开始时间，但并不适合我们所设想的情况，因为动画标题一开始并没有颜色。动画开始之前，用户根本看不到蓝色背景，或者说根本就没有任何背景色。

更复杂的玩法

在本例中，我们在每个关键帧中只变更了背景色，实际上只要你愿意，可以在里面增加任意多个属性。将它们如其他 CSS 规则一样写进同一个关键帧的花括号里。比如说，如果你想同时改变字号和背景色，可以这样做：

```
@-webkit-keyframes fade {
    0% {
        background: hsla(203,78%,36%,.2);
        font-size: 100%;
    }
    100% {
        background: none;
        font-size: 120%;
    }
}
```

一个元素可以设置多个动画，因此你可以将上面的动画分成两个独立的动画：

```
@-webkit-keyframes fade {
    0% {background: hsla(203,78%,36%,.2);}
    100% {background: none;}
}
@-webkit-keyframes scaletext {
    0% {font-size: 100%;}
    100% {font-size: 120%;}
}
```

然后在同一个元素上设置两个动画：

```
h2:target {
    -webkit-animation-name: fade, scaletext;
    -webkit-animation-duration: 2s;
    -webkit-animation-iteration-count: 1;
    -webkit-animation-timing-function: ease-in;
}
```

将动画分开将产生更多的代码，但在复杂的动画里可以便于你跟踪每一个步骤，而且还可以独立地为每个动画设置持续时间、播放次数和其他属性。还有一个优点是你可以将动画复用到其他元素上。比如，你可能想为 `h2` 设置“淡出”和“字号变化”两个动画，但只想为 `h3` 设置“字号变化”这一个。既然动画都是独立的，你就可以复用“字号变化”而不用为 `h3` 重新书写一个。

`animation` 属性的来龙去脉

`animation` 属性是动画（Animation）模块的一部分，你可以在 www.w3.org/tr/css3-animations 找到它。它是 `animation-name`、`animation-duration`、`animation-timing-function`、`animation-delay`、`animation-iteration-count`、and `animation-direction` 的简写属性（有顺序要求）。

在使用上述属性为某个元素创建动画之前，你需要首先为动画命名，并使用`@keyframes`规则定义动画的内容。关键帧是动画过程中的多个时间点，用百分数表示；关键词 from 和 to 分别等同于 0%和 100%。每个关键帧包含了适合应用时的样式规则。浏览器从首个关键帧往后逐步变更样式。

除了用来使背景色淡出之外，你还可以将 `animation` 用在以下场景。

- ❑ 实现按钮有节奏地发光，请参阅www.zurb.com/playground/radioactive-buttons。
- ❑ 读取状态提示，请参阅http://24ways.org/2009/css-animations。
- ❑ 闪视卡（Flash Card），请参阅 http://line25.com/articles/super-cool-css-flip-effect-with- webkit-animation。
- ❑ 元素滚入界面的效果，请参阅www.zurb.com/playground/sliding-vinyl。

网上有许多更复杂，看起来更像电影的 CSS 动画效果，我列在这里的仅是一些更具有实践意义的例子。弄清楚它究竟能实现些什么效果是很有趣的，请上 Google 搜索“CSS3 Animation”试试。

表5-6 **`animation`** 属性的浏览器支持度

IE	Firefox	Opera	SafariI	Chrome
不支持	不支持	不支持	支持，需要`-webkit-`前缀	支持，需要`-webkit-`前缀

除了使用 `animation-delay` 来延迟动画开始的时间之外，我们还可以在动画中增加一个关键帧来让背景保持为暗蓝色：

```
@-webkit-keyframes fade {
    0% {background: hsla(203,78%,36%,.2);}
    20% {background: hsla(203,78%,36%,.2);}
    100% {background: none;}
}
@keyframes fade {
    0% {background: hsla(203,78%,36%,.2);}
    20% {background: hsla(203,78%,36%,.2);}
    100% {background: none;}
}
```

改动之后，动画一开始会立即显示一个蓝色背景，保持这个颜色直到 20%的进度（0.4 秒），接着开始淡出直到变成透明。

说明 想要体验动画的效果，请用 Safari 或 Chrome 打开 target_5.html 文件，该文件包含在本章的示例文件里，它包含了至此为止的所有修改。

5. 针对其他浏览器的兼容方案

淡出的动画目前在 Safari 和 Chrome 上运行正常了，但其他浏览器怎么办？为了保证动画的正常运行，我们必须从当前聚焦的 `h2` 元素上移除背景色 ，因此现在你在非 Webkit 内核的浏览

器上点击目录里的链接后，它们仍会保持透明不变。

我们所需要的方案是让不支持 CSS 动画的浏览器识别第一条 `h2:target` 规则，而支持该特性的浏览器则识别应用了透明背景色的第二条规则。我们可以使用早在第 1 章就介绍过的 Modernizr 脚本。

该脚本已包含在你下载的本章示例文件中，所以请你直接在页面的 `head` 中加上如下代码：

```
<script src="scripts/modernizr-1.6.min.js"></script>
```

说明　本书编撰之时，示例文件中所包含的 Modernizr 版本是 1.6。但当你阅读时它可能已经有了最新的版本，请你下载并予以替换。可以到 www.modernizr.com 查看版本。

该脚本在页面的 `html` 元素中增加类来表示相关特性是否被支持。举个例子，Modernizr 会为 Safari 和 Chrome 浏览器在 `html` 标签上增加一个名为“cssanimations”的类，而所有其他浏览器将会被赋予另外一个名为“no-css-animations”的类。我们可以改变第二个 `h2:target` 选择器，让它仅在 cssanimations 类存在时方才有效：

```
.cssanimations h2:target{
    padding: 0;
    background: none;
    -webkit-animation: fade 2s ease-in 1;
    animation: fade 2s ease-in 1;
}
```

保存页面，然后分别用一个 Webkit 内核浏览器和一个非 Webkit 内核浏览器查看它。在前一个浏览器上，当标题被聚焦时你将看到运行中的淡出动画。而在后者上，当标题被聚焦时你将看到淡入的过渡。而在既不支持过渡和动画特性，却支持`:target`（比如 IE 9 和 Firefox 3.6 及其之后的版本）的浏览器上，当标题被聚焦时你将看到蓝色的背景色立即展现出来。而最后，对于不支持以上所有特性的浏览器（比如 IE 8 及之前的版本），当你点击目录中的链接时并不会展现任何特效。

当然了，目录的链接依旧可用——它们会像往常那样把用户带到页面的下方，因此使用低端功能浏览器的用户将不会察觉到他们错过了什么内容。如果你真的必须在这些浏览器下实现淡出效果，你可以选择如下解决方案。

- ❑ 使用动态的 GIF 图片。诚然，这个方法有些过时，但它同时也是最完美的解决方法。但请注意这个方法只适用于那些支持`:target` 伪类的浏览器。你所需要做的，只是制作一个颜色变化的动态 GIF，接着把它设置为 `h2:target` 的背景图。可以阅读一篇完整地介绍该方法的文章“Star on :target”（http://thinkvitamin.com/dev/stay-on-target），作者是 Brian Suda。
- ❑ 使用脚本。无论浏览器是否支持`:target` 伪类该方案都有效（当然，用户必须启用了 JavaScript，而且该脚本需要支持用户的浏览器）。如何编写这个脚本已经超出了本书涉及

的范围，但你可以访问 www.marcofolio.net/css/css3_animations_and_their_jquery_equivalents.html 和 http:// weston.ruter.net/projects/jquery-css-transitions 来获取一些脚本，它们可以兼容所有的过渡和动画效果。或者别忘了上 Google 搜索“Javascript Animation Framework”。最后，如果你还对淡出技术感兴趣，请用 Google 搜索“Yellow Fade Technique”，你将找到大量的脚本——其中一些是独立的，另一些则依赖于某个特定框架——这就留给你自己去抉择了。

但无论使用上述哪个选项，请记得使用 IE 条件注释或 Modernizr 对不同浏览器隐藏掉额外的 CSS 或脚本。

第 6 章

不同的屏幕尺寸，不同的设计

网页的浏览终端越来越多样化早已不是什么秘密，也没人会因此而感到惊讶。用户可以通过宽屏电视、台式电脑、笔记本、手机甚至是一台冰箱来访问你的网站。尽管你无法保证一个网站在不同屏幕尺寸和不同字号的设备上看起来都完全一样，但至少可以让网站去适配用户的设置，使它在可用的屏幕空间里有良好的展现和完整的功能。在本章中，你将会学到如何使用 CSS3 的“媒体查询”（Media Queries）特性来让一个页面设计快速地适应于不同的屏幕尺寸，从而使你的页面显得更高效、灵活和易用。

你将学到

我们将使用 CSS3 重新编写一个页面布局以使之适应不同的屏幕尺寸和设备：

- 使用“媒体查询”特性根据访客的设备属性分别应用不同的样式；
- 使用“多列布局”使文字自适应于并列的栏。

6.1 基础页面

图 6-1 展示出一个虚拟面包坊的页面布局。该布局是一个液态布局（也称为“流态布局”），它可以根据浏览器窗口的宽度进行自我调整，从而适用于不同的屏幕尺寸而不会出现横向滚动条或引起某些元素重叠。然而在某些屏幕尺寸上它理应比其他情况的页面看起来更好一些。在极宽或极窄的窗口上，该设计都可以正常使用并且看起来尚可接受，只是并没有 800px 至 1200px 范围内看起来那么吸引人（见图 6-2）。

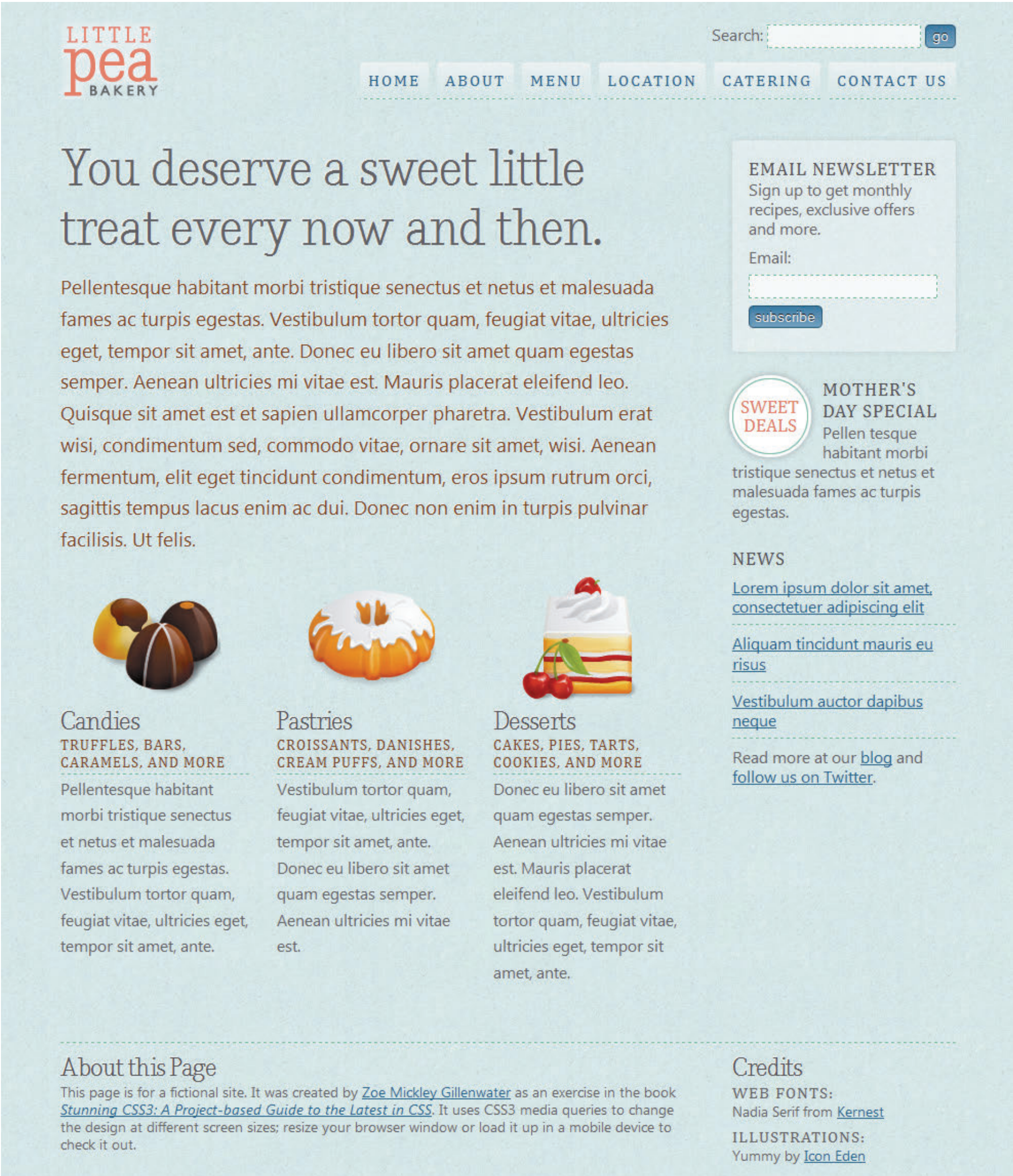


图 6-1　虚拟的“小豆子面包坊”主页在 1024px 宽的浏览器窗口上的表现

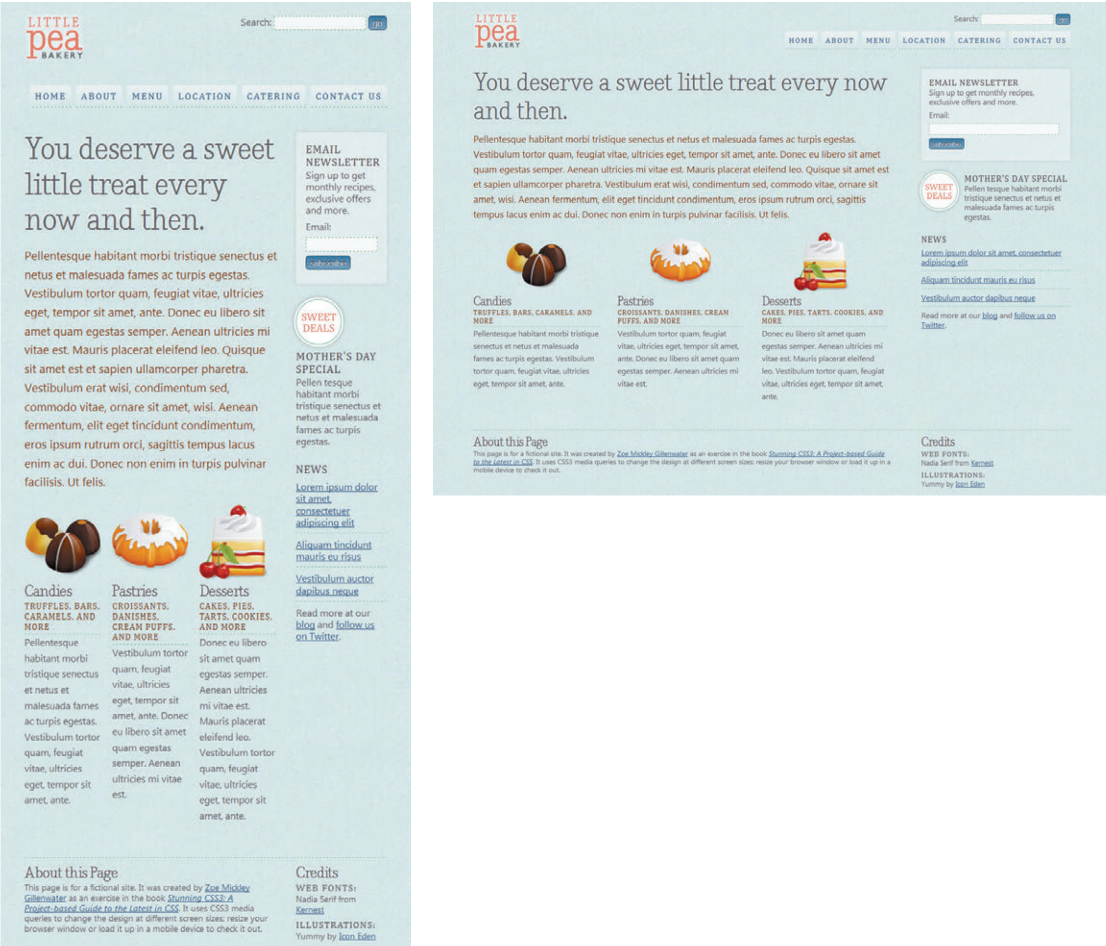


图 6-2　页面在极宽和极窄的窗口上看起来尚可，但并不出色

在《灵活 Web 设计：使用 CSS 创建液态和弹性布局》一书的第 2 章中，我向大家展示了液态布局如何在保证设计的美妙和丰富的同时，又能适应多种屏幕尺寸 。你可以选用灵活的图片、适当的文本长度以及创造性地使用页面留白来保证设计适用于很大的宽度范围。然而我并不否认一点，即制作出一个在 2000px 时每个部分都被精确实现的页面而且将它压缩至 300px 之后依旧能够保持完美，几乎是不可能的。一直以来我都提倡使用 `min-width` 和 `max-width`，并为移动设备编写不同的样式来解决这个问题。

《灵活 Web 设计》一书动笔于 2008 年，然而在那之后，一款用来创建页面布局的新式工具已经有了很好的浏览器支持度，它可以使页面在任意你想要的尺寸范围内工作，这个工具就是：媒体查询。

6.2　媒体查询是什么

媒体查询可以让你根据用户所使用的设备或显示器特性来自定义样式，比如可视区域的宽度，目前处于纵向还是横向模式[①]，或者是否支持彩色显示等。这与 CSS 2.1 里的诸如 `screen` 和 `print` 等媒体类型（Media Type）的概念不同。使用媒体查询，你不仅可以针对特定的媒体类型设置一系列样式，还可以针对用户显示终端的一个或多个特性来定义样式。下面是一个例子：

```
@media screen and (max-width: 600px) {
    body {
      font-size: 88%;
    }
    #content-main {
      float: none;
      width: 100%;
    }
}
```

> **说明**　所有可用的媒体特性都罗列在 www.w3.org/tr/css3-mediaqueries/#media1 里，并包含详细释义。你可能最常需要用到的是以下特性：`min-width`、`max-width`、`min-device-width`、`max-device-width`、`orientation`（横向或纵向）、`color` 和 `resolution`。

上面的媒体查询以`@media` 规则开始，然后指定了媒体类型（本例中是 `screen`）。接下来的 `and` 之后跟着我们想要匹配的特性，称作媒体特性（Media Feature）。我们所指定的媒体特性 `max-width: 600px` 告诉浏览器：以下用花括号嵌套起来的一整段样式都是针对宽度为 600px 之内的可视区域的。如果可视区域的宽度超过了 600px，那么浏览器将忽略这段媒体查询的样式。

> **说明**　如果想要在媒体查询里使用 `all` 这个媒体类型，可以完全去掉它和连接词 `and` 使代码更短，就像这样：`@media (max-width:600px)`。

媒体查询代码可以直接写进你的主样式文件，这样能保证所有的样式都在同一处，便于调试和维护，也能减少 HTTP 的请求。如果你愿意，也可以用它配合 `link` 元素和`@import` 规则来引入不同的样式文件。

```
@import url(narrow.css) only screen and (max-width:600px);

<link rel="stylesheet" media="only screen and
¬ (max-width:600px)" href="narrow.css">
```

① 也被译成“肖像模式”和“风景模式”。

这里，我在媒体类型 screen 之前加入了关键字 only 来防止某些稍旧的浏览器可能因为不支持媒体查询而将样式全部载入。大多数不支持该特性的浏览器将不会使用该样式文件，但这样做只是为了多一层额外的保障。如果你选择将代码直接写入主样式文件，则无需使用关键字 only。

不管是独立成额外的样式文件，还是和其他 CSS 混写在一起，媒体查询对于 Web 设计来说都是一款功能强大的工具。借助它，我们可以比以往任何时候都更精确地针对用户的设备和设置来自定义和调优样式。这能同时提升页面的吸引力和可用性。我们可以更改文本的长度、行距和字号来保证它们在不同的宽度下都便于识别；我们可以针对窄小的屏幕重新排列栏目，缩放甚至移除图片以更好地利用页面留白，并让用户准确获得他们想要的内容；我们还可以为触屏的移动设备放大链接来使它们更方便用户点击。要做到上述一切，我们不用依赖复杂的脚本嗅探浏览器、探测特性或切换样式文件。只需针对不同的应用场景编写不同的 CSS 即可，而用到的都是你目前已掌握的知识。

接下来让我们一起在示例文件中运用媒体查询特性设计一个适用于宽屏、窄屏和移动设备的页面。

6.3　针对宽屏来改变布局

我们首先针对宽屏终端编写样式。开始之前，请到 www.stunningcss3.com 下载本章的示例文件并用编辑器打开 media-queries_start.html 页面。其 CSS 的定义包含于页面 head 里的一个 style 元素里。

示例页面在 1200px 左右的宽度会有一些拉伸错位，因此我们新增一个仅针对 1200px 及以上宽度的媒体查询代码。请将如下 CSS 代码加入到 head 区域的 style 元素中：

```
@media screen and (min-width: 1200px) {
}
```

以上媒体查询代码必须放置在所有样式的后面，以便它能覆盖之前的级联样式设置。它告诉浏览器我们希望针对 screen 媒体类型应用这个媒体查询之内的样式，但仅当用户的可视区域宽度大于等于 1200px 时才有效。当然，目前这个媒体查询里并没有任何样式，只是一对等待填充内容的花括号。既然在 1200px 的宽度下我们有如此大的空白区域，何不将布局由两栏变成三栏呢？

为了实现这个设想，我们需要更改导航条 div 的位置，还有两个内容 div 的宽度和外边距。以下便是当前上述三个 div 的样式代码，它们处于媒体查询区块之外：

```
#nav-main {
   float: right;
   margin: 40px 0 0 0;
}
#content-main {
```

```
    overflow: hidden;
    float: left;
    width: 70%;
    margin-bottom: 40px;
}
#content-secondary {
    float: right;
    width: 25%;
    margin-bottom: 40px;
}
```

在我们刚才新增的媒体查询区块里增加新的样式代码，以适应宽度超过 1200px 的情况：

```
@media screen and (min-width: 1200px) {
    #nav-main {
      position: fixed;
      top: 136px;
      width: 13%;
      margin: 0;
    }
    #content-main {
      width: 58%;
      margin-left: 18%;
    }
    #content-secondary { width: 20%; }
}
```

说明 Opera 10.6 上有一个奇怪的问题：首先，当你将窗口拉伸至 1200px 以上时，导航条的 div 会消失；而当你将鼠标移上导航条本该出现的区域时，它又冒了出来。针对这个问题，目前还没有解决方案，希望 Opera 开发团队可以尽快修复该问题。

这个改动会将导航条 div 定位到 Logo 下方，为创建第三栏而做准备。为了给它腾出空间，ID 为 content-secondary 的 div 宽度需要从 25%下降至 20%；而 ID 为 content-main 的 div 宽度则需要从 70%缩减至 58%，接着再给它增加左外边距。

让我们将页脚区域的关于（About）和贡献（Credits）div 的宽度做同步调整。把它们的 ID 加到媒体查询里#content-main 和#content-secondary 的规则之后：

```
#content-main, #about {
    width: 58%;
    margin-left: 18%;
}
#content-secondary, #credits { width: 20%; }
```

至此，所有的页面元素都已更好地定位以适应更大的宽度（见图 6-3）。保存页面，并用尚不落伍的浏览器打开预览。拉伸窗口直到超过 1200px，你可以看到页面布局是如何自动地适应宽度变化的。

6

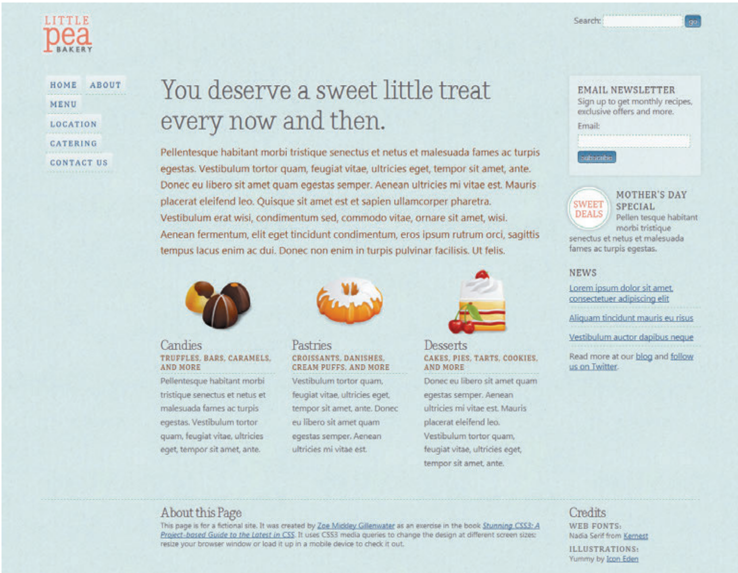


图 6-3　页面中的元素更好地利用了较宽窗口的空白区域

媒体查询的来龙去脉

媒体查询（Media Queries）是其同名模块的一部分，你可以在 www.w3.org/tr/css3- mediaqueries 找到它。借助它你不仅可以根据 `screen` 和 `print` 等媒体类型自定义样式，还可以依据用户显示终端的特性（如可视区域的宽度）来设置样式。这些特性被称为“媒体特性”，罗列在 www.w3.org/tr/css3-mediaqueries/#media1 中；并非所有支持媒体查询的浏览器都支持全部的特性。

一个媒体查询可以定义在一个样式文件中，它使用`@media`规则，后跟媒体类型和一个（含）以上的媒体特性。除了`@media`之外，媒体查询还可以用于`link`元素和`@import`规则。

一个媒体查询可以包含一个或多个媒体特性，比如`@media screen and (min-width: 320px) and (max-width:480px)`。你也可以对同一个`@media`规则使用多个用逗号分隔开来的媒体查询，比如`@media screen and (color), projection and (color)`，这一点和选择器分组类似。

如果你希望在某个媒体查询结果为非真时就应用某些样式，可以在媒体查询之前加上单词`not`，比如`@media not print and (max-width:600px)`。

除了针对不同屏幕尺寸更改界面布局以外，你还可以将媒体查询用在以下场景。

- 根据不同的文字行长度调整字号和行距以确保文字更易读，请参阅 http://forabeautifulweb.com/blog/about/proportional_leading_with_css3_media_queries。
- 为移动设备增加按钮、标签和链接的字号以方便你的手指在触摸屏上更易触发这些元素。
- 针对窄屏移动设备减小字号，因为用户界面实际上已经被放大显示并使得文字看起来比桌面屏幕上所展现的文字更大。

- 明确告诉小屏移动设备用户页内链接会跳转到同页面的下方。
- 为诸如 iPhone4 之类的高分辨率设备改用高清图片，请参阅 http://dryan.com/articles/posts/2010/6/25/hi-res-mobile-css-iphone-4；也可参考本章后续的示例。
- 为不同的可视区域显示不同尺寸的图片。
- 为不同尺寸的纸张应用不同的打印样式。

表 6-1 媒体查询的浏览器支持度

IE	Firefox	Opera	Safaril	Chrome
部分支持，9+	部分支持，3.5+	部分支持	部分支持	部分支持

我将所有的浏览器评定为“部分支持”是因为它们并不支持所有已存在的媒体特性。具体的浏览器支持情况由于受篇幅限制就不再详细列出，然而可以肯定的是上述浏览器都支持大多数的媒体特性，而且包括你将会经常使用的那部分。

6.3.1 由横向导航条变为纵向目录

尽管目前所有的元素都已定位到我们所设想的位置，但其中一些页面元素还可以做进一步地装饰。比如说，导航条 `div` 里的 `li` 元素已设置成浮动并有了左外边距，这样做是为了使它们水平对齐并保持彼此分离，但这也使它们无法做到每个一行地衔接起来，因为我们想要实现一个纵向的目录。它们还有一些轻微的顶部圆角，这个效果在横向排列时很不错，但当它们彼此顶部和底部相连时就不那么好看了。现在当我们需要将链接装扮成纵向的目录时就不再需要这些样式了，因此我们在媒体查询里用新的样式去覆盖它们：

```
#nav-main li {
    float: none;
    margin: 0;
}
#nav-main a {
    -moz-border-radius: 0;
    -webkit-border-radius: 0;
    border-radius: 0;
}
```

现在每个链接都另起一行并占据了目录的全部宽度（见图 6-4）。



图 6-4 现在每个链接都占据了目录的全部宽度

接下来，让我们把目录作为整体增加一些样式，好让它的风格看起来与页面另一边的邮件订阅栏更统一，即拥有半透明的背景、轻微的圆角和一点很淡的投影：

```
#nav-main {
    position: fixed;
    top: 136px;
    width: 13%;
    margin: 0;
    -moz-box-shadow: 0 0 8px hsla(0,0%,0%,.1);
    -webkit-box-shadow: 0 0 8px hsla(0,0%,0%,.1);
    box-shadow: 0 0 8px hsla(0,0%,0%,.1);
    -moz-border-radius: 3px;
    -webkit-border-radius: 3px;
    border-radius: 3px;
    background: hsla(0,0%,100%,.3);
    text-align: right;
}
```

由于目录已经有它自己的背景色，现在需要调淡链接自己的半透明渐变，这样才不至于因为两种颜色的层叠而变得太不透明：

```
#nav-main a {
    -moz-border-radius: 0;
    -webkit-border-radius: 0;
    border-radius: 0;
    background: -moz-linear-gradient(hsla(0,0%,100%,.3),
    ¬ hsla(0,0%,100%,0) 15px);
    background: -webkit-gradient(linear, 0 0, 0 15,
    ¬ from(hsla(0,0%,100%,.3)), to(hsla(0,0%,100%,0)));
}
#nav-main a:hover {
    background: -moz-linear-gradient(hsla(0,0%,100%,.6),
    ¬ hsla(0,0%,100%,.2) 15px);
    background: -webkit-gradient(linear, 0 0, 0 15,
    ¬ from(hsla(0,0%,100%,.6)), to(hsla(0,0%,100%,.2)));
}
```

这些更改完成了由水平导航条向纵向目录的转变（见图 6-5）。



图 6-5　左侧边栏的目录样式已参考右侧的邮件订阅栏进行了重新定义

6.3.2　多列文本

界面根据可视区域的宽度调整之后，用户的主要抱怨之一便是文本行的长度不是太短就是太长，从而影响阅读体验或降低阅读欲望。这种对于“非标准的”文本行长度的恐惧其实是基于假

设和猜想，在现实中并不存在适用于所有人的魔法值，一个用户的年龄、阅读级别、母语、是否残障等因素均会影响他对最适合的文本行长度的看法。

说明　你可以在《灵活 Web 设计》的第 1 章中学到更多有关文本行长度的知识，并可以从 www.flexiblewebbook.com/bonus.html 免费下载。

然而，文本行的长度在极端的边界条件下并不适合大多数的读者，而且也并非永远都看起来那么具有吸引力。我们目前可以用来控制该值的办法之一是使用 CSS3 所新增的多栏属性。这些属性可以帮助你将一个 HTML 元素变成多栏，就像是报纸的排版那样。

你可以用 `column-count` 或 `column-width` 来创建栏目，在使用后者时，浏览器将根据目前的可用宽度来确定有多少个栏目（你也可以同时使用以上两个属性，但可能会遇到未知的问题；请阅读后面"多栏属性的来龙去脉"来获得更多相关信息）。

让我们把常用布局和宽屏布局的引言段落变更为两栏。请在媒体查询区块的外面找到现有的 `h1 + p` 规则；它应该在第 102 行，大致在 `style` 元素里的第三个部分。现在为其增加一个 `column-count` 属性，外加三个浏览器私有版本：

```
h1 + p {
   -moz-column-count: 2;
   -o-column-count: 2;
   -webkit-column-count: 2;
   column-count: 2;
   color: #7F4627;
   text-shadow: -1px -1px 0 hsla(0,0%,100%,.6);
   font-size: 120%;
}
```

目前还没有浏览器支持无前缀版的 `column-count` 属性，而且 Opera 由于不支持多栏特性，所以也不存在`-o-column-count` 这个属性，但为了向后兼容添加以上两个属性是一个明智的选择。

该属性告诉基于 Mozilla[①]和 Webkit 内核的浏览器你希望将段落分隔成两栏，但这些栏目的盒容器并不是 HTML 文档树里真实存在的元素，而是浏览器用来填充内容的虚拟盒容器。该段落目前被 W3C 称作*多栏元素*（Multicol Element）——它是一个多列布局的容器。

你可以用 `column-gap` 属性来控制栏目之间的间隔。现在给 `h1 + p` 规则设置为 `1.5em`：

```
h1 + p {
   -moz-column-count: 2;
   -moz-column-gap: 1.5em;
   -o-column-count: 2;
   -o-column-gap: 1.5em;
   -webkit-column-count: 2;
   -webkit-column-gap: 1.5em;
   column-count: 2;
   column-gap: 1.5em;
   color: #7F4627;
```

① Mozilla 出品的浏览器所使用的内核名叫 Gecko。

```
        text-shadow: -1px -1px 0 hsla(0,0%,100%,.6);
        font-size: 120%;
    }
```

如果你没有设置 column-gap 属性值，则每个浏览器将分别为其指定默认值，因此最好还是由你自己进行明确地设置。在这里我们使用了 em 单位值，因此栏目间的空隙将随着文字大小的变化而变化，从而使文字更易阅读。

现在，引言段落已被分隔成两栏，不仅应用于普通布局，也利用媒体查询将其应用于宽屏的布局（见图 6-6）。至此就完成了所有针对宽屏版面包坊页面的样式定义（见图 6-7）。



图 6-6　引言段落的文本在 Firefox、Safari 和 Chrome 上变成了两栏

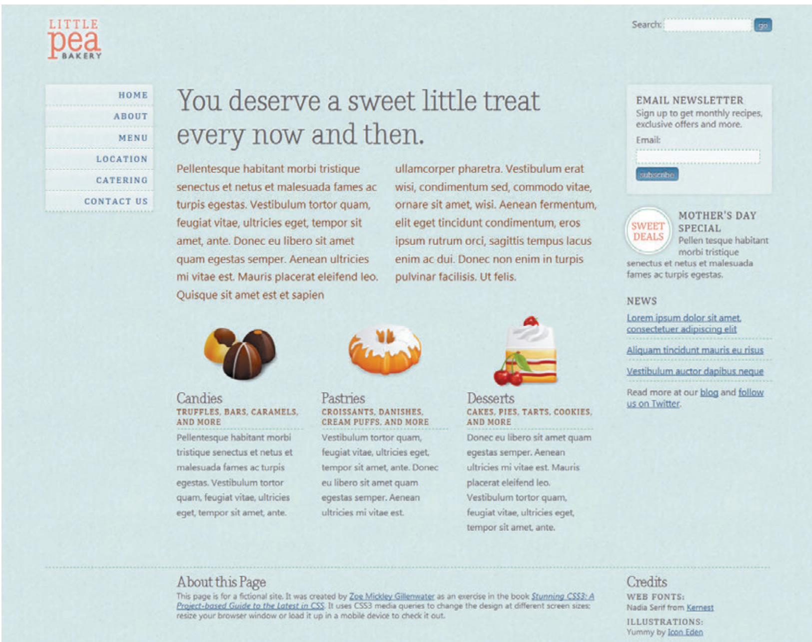


图 6-7　宽屏版的设计成品

说明 至此为止的所有修改都在 `media-queries_1.html` 页面里，它在你所下载的本章的示例文件之中。

---

1. 多栏所存在的问题

尽管在我们的面包坊页面的引言段落里 CSS3 多栏表现良好，但它仍存在一系列影响其实用性的问题，你应该在使用该特性之前时刻注意这些问题。其中一些问题是纯粹的技术问题，因此随着 W3C 完善规范和浏览器修复缺陷并增强支持度，这些问题会逐步削减甚至消失。它们包括如下内容。

- **均衡分配栏目的高度**。如果没有足够多的内容用来均匀地填充每个栏目，那么浏览器将面临哪个栏目会得到更多高度的抉择。不同的浏览器选择的结果也不同，而这有时就会引起不可预料的结果。
- **多栏共享相同的内、外边距和边框**。基于 Webkit 内核的浏览器会让内、外边距和边框的值被多个栏目共享，这会造成一个非常奇怪的界面。
- **内容被错误地分栏**。能够控制内容分隔的位置是很重要的，比如你希望某个标题与其内容呆在一起，可以使用 `column-break` 属性实现这个需求，但目前没有任何浏览器支持它。
- **处理栏目或内容溢出**。当内容或栏目无法完全符合容器（即多栏元素）时会发生溢出，但目前众多浏览器还无法就处理方式统一；它可能会向右或向下溢出，或是直接截断。如果某块独立的内容对于某个栏目来讲显得过大（比如一张比栏目更宽的图片），那么它理应直接在栏目的分隔处截断，但是 Firefox 会让图片溢出，而 Webkit 则会在栏目边缘将其截断，并非如规范中所定义的那样。
- **栏目内的浮动内容**。一个多栏元素内的浮动理应相对于它们所出现的栏目来定位，但 Webkit 很诡异地完全在多栏元素之外浮动。
- **打印时的页码处理**。打印页面时，如果一个多栏元素需要多页才能打印完，那么分栏内容就不应跨页断开。即：内容必须从第一页开始进行分栏打印，然后内容在次页被衔接起来继续分栏打印，直至内容打印完成。早期的 Webkit 内核浏览器并不遵守这个规范，但目前版本的 Webkit 将分栏内容退化成一整个通栏内容进行打印，从而完全避开了这个问题。

说明 列表中的前两个问题可以参考如下网址：http://zomigi.com/blog/deal-breaker-problems-with-css3-multi-columns。

然而，一些多栏的问题更多源于在网页上实现多栏这个想法本身。读者需要向下滚屏来阅读完一个栏目，然后返回到上面继续读下一个栏目，如此反复——多少显得单调无聊。如果分栏后的内容高于屏幕的可视区域，这就不是一个技术缺陷——而是可用性上出现了问题。将网页当做

印刷品来对待通常并不会获得很好的体验。如果想了解更多因CSS3多栏而产生的可用性或设计问题，请阅读Roger Johanssen的“Multicolumn layout considered harmful”（www.456bereastreet.com/archive/200509/css3_multicolumn_layout_considered_harmful）、Richard Rutter的“More on multi- column layouts”（www.clagnut.com/blog/1590）和Andy Clarke的“CSS3 Multi-Column Thriller”（www.stuffandnonsense.co.uk/archives/css3_multi-column_thriller.html）。

由于有如上的这些问题，我强烈建议大家有限度地使用多栏特性。举个例子，我认为对于一两个段落或一个列表，使用多栏是没问题的。但我并不认为它们适用于某个与页面等高的区块或是包含诸多段落、元素类型或图片的复杂内容。记住我的建议，然后明智地使用多栏特性。

2. 针对其他浏览器的兼容方案

多栏是一项典型的渐进增强技术，假如浏览器不支持多栏属性，那么将会直接看到文本的最初排版——单栏。

如果你非要为尚不支持该特性的浏览器找到解决方案，有几款脚本可以将内容分为多栏。其中，Cédric Savarese开发的CSS3 Multi Column脚本（www.csscripting.com/css-multi-column）是很不错的一个，因为它会从CSS里读取已经设置过的多栏属性值，然后使其支持不支持该特性的浏览器。你还可以从如下脚本中进行选择。

- Columnizer jQuery插件，作者Adam Wulf（http://welcome.totheinter.net/columnizer-jquery-plugin）。
- MooColumns MooTools类，作者Jason J.Jaeger（http://greengeckodesign.com/moocolumns）。
- Multi-column脚本，作者Randy Simons（http://randysimons.nl/125,english/129,multi-column-text）。
- Column脚本，作者Michael van Ouwerkerk（http://13thparallel.com/archive/column-script）。

多栏特性的来龙去脉

多栏（multi-columns）是多列布局模块的一部分，可以在www.w3.org/tr/css3-multicol找到它。可以用`column-count`或`column-width`其中之一（或同时使用）来创建多栏。也可以用`columns`这个简写属性同时设置上述两个属性值，但Firefox并不支持。

`column-width`属性可以让浏览器根据页面的可用空间来决定分成多少个栏目。你为`column-width`设置的值实际上更像是其最小值；比如，你对一个`250px`宽的容器设置其`column-width`属性值为`100px`，但将`column-gap`设置为`0`，那么浏览器将生成两个`125px`宽的栏目。

`column-count`属性则能让你指定栏目的精确数量，其宽度则根据可用空间自动计算。如果同时设置了`column-count`和`column-width`，那么`column-count`值就被当做栏目的最大值。举个例子，同样对于`250px`宽的一个容器，如果你设置`column-width`为`100px`，`column-count`为`3`，那么浏览器将生成两栏而不是三栏。

你可以使用`column-gap`属性设置栏目间隔，并使用`column-rule`属性给每个间隔绘制一条竖线来实现视觉分隔。`column-span`属性则可以让元素跨越多栏，但目前尚无浏览器可以支持。

`break-before`、`break-after` 和 `break-inside` 三个属性可用来控制内容分栏的位置，但它们还尚未被任何浏览器支持。

除了如本章一样将小段的文本分隔成多栏以外，我认为对于多栏特性，唯一安全的用法就是将一个包含少量项目的独立列表分隔成多栏；请参阅：http://trentwalton.com/2010/07/19/css3-multi-column-layout-column-count。

表 6-2 多栏特性的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	部分支持	不支持	部分支持	部分支持

6.4 针对窄屏来改变布局

针对宽屏的布局调整完成以后，现在让我们把目光投向窄屏终端。首先，我们要在之前增加的第一个媒体查询的下方另外新增一个，其目标是可视区域窄于 760px 的终端：

```
@media screen and (max-width: 760px) {
}
```

这将告诉浏览器我们希望填加到媒体查询之内的样式是应用于媒体类型 `screen` 而可视区域小于 760px 的终端。我为何要选择 760px 这个宽度？因为该宽度值可以防止这些样式被应用于 800×600 分辨率的桌面显示器的最大化窗口，还包括 768×1024 分辨率的 iPad。我希望这些终端能够应用普通的样式，因为在上述分辨率下界面能够保持不错的状态。但小于 760px 的宽度，界面就开始看起来有些拥挤，内容溢出容器的可能性也随之增加。

同样地，让我们先拿导航条开刀——修改样式让它更好地符合可用的宽度。当窗口宽度过窄时，整个导航条会另起一行并排列到 logo 的下方，这并无大碍；它仍会保持向右对齐，但当左方没有了 logo 之后看起来效果不佳。因此当它在 logo 下方另起一行时，我们就把导航条改为向左对齐：

```
@media screen and (max-width: 760px) {
    #nav-main {
      clear: left;
      float: left;
    }
    #nav-main li { margin: 0 .5em 0 0; }
}
```

你可以从图 6-8 看出修改前后的差别。这是一项细微的改动，但效果却很不错，而且只需要书写一点点 CSS 便可以实现。



图 6-8　导航条在窄小的窗口下无法位于 logo 旁边，因此我们使用媒体查询将其由右对齐改成左对齐

接下来，让我们去除引言段落的分栏效果——当窗口宽度小于 `760px` 时，分栏看起来非常难看（见图 6-9）。在第二个媒体查询区块里书写一条新的 `h1 + p` 规则，并在其中将栏目数改为 1：

```
h1 + p {
   -moz-column-count: 1;
   -o-column-count: 1;
   -webkit-column-count: 1;
   column-count: 1;
}
```



图 6-9　在窄小的窗口下，一个栏目每行仅能显示两三个英文单词

改动之后，引言段落里每行的长度看起来合理多了（见图 6-10），但在它正下方并列显示的三个段落仍旧看起来非常窄小（见图 6-11）。我们将继续予以改进。

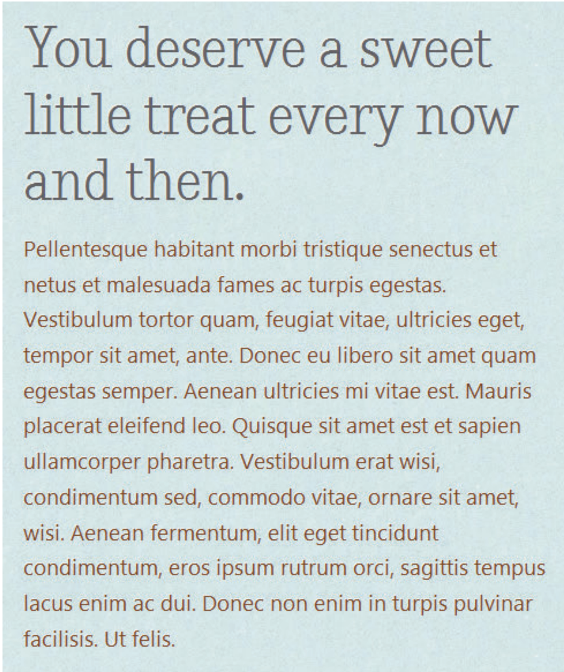


图 6-10 在窄小的窗口下将 `column-count` 的值减少到 1 使得引言段落既美观又更便于阅读



图 6-11 特色产品框在窄屏下显得非常窄小，只得紧靠在一起

目前，每个特色产品框都是在左侧浮动的一个 div。取消浮动将使它们改为由上至下排列，而且占据主内容 div 的整个宽度。当它们由上至下排列时，位于每个上部的图片就显得不是非常美观，而将图片移到左边来就显得更为合理。因此我们给媒体查询增加如下的新规则：

```
.feature {
   float: none;
   width: auto;
   margin: 0 0 1.6em 0;
   padding: 0 0 0 140px;
   background-position: top left;
}
```

说明　三张图片都来自 Icon Eden 出品的 Yummy 免费图标集（www.iconeden.com/icon/yummy-free-icons.html）。

这条规则取消了特色产品框的浮动和宽度设置，并为了给图片腾出空间将上内边距替换成了左内边距（因为使用的是背景图），以便让它显示于产品框的左侧。



图 6-12　将特色产品框层叠起来并将图片改为左侧显示之后，它在窄小窗口的有限空间里看起来好多了

至此，对窄屏版页面还剩下最后一项修改：目前右侧的栏目太窄了，增加了其内容中的长单词溢出的可能性，栏目标题便是其中最“危险”的一个，因为它全是大写字母从而占据了更多的位置。我们可以通过减小字号和文字间距来降低其溢出的风险：

```
h3 {
    font-size: 100%;
    letter-spacing: 0;
}
```

你可以看到这条规则所产生的效果（见图 6-13）。这同样是一次细微的修改，却降低了文字溢出的风险，也使文字更方便阅读了。



图 6-13 减小侧栏标题的字号和字间距降低了窄小窗口下文字溢出的可能性

至此我们就完成了针对窄屏版面包坊页面的所有修改（见图 6-14）。保存页面并在尚不落伍的浏览器中打开，缩放窗口便可查看页面在极窄和极宽条件下的设计变化。

说明 至此所有修改都在 media-queries_2.html 页面里，它在下载本章示例文件之中。

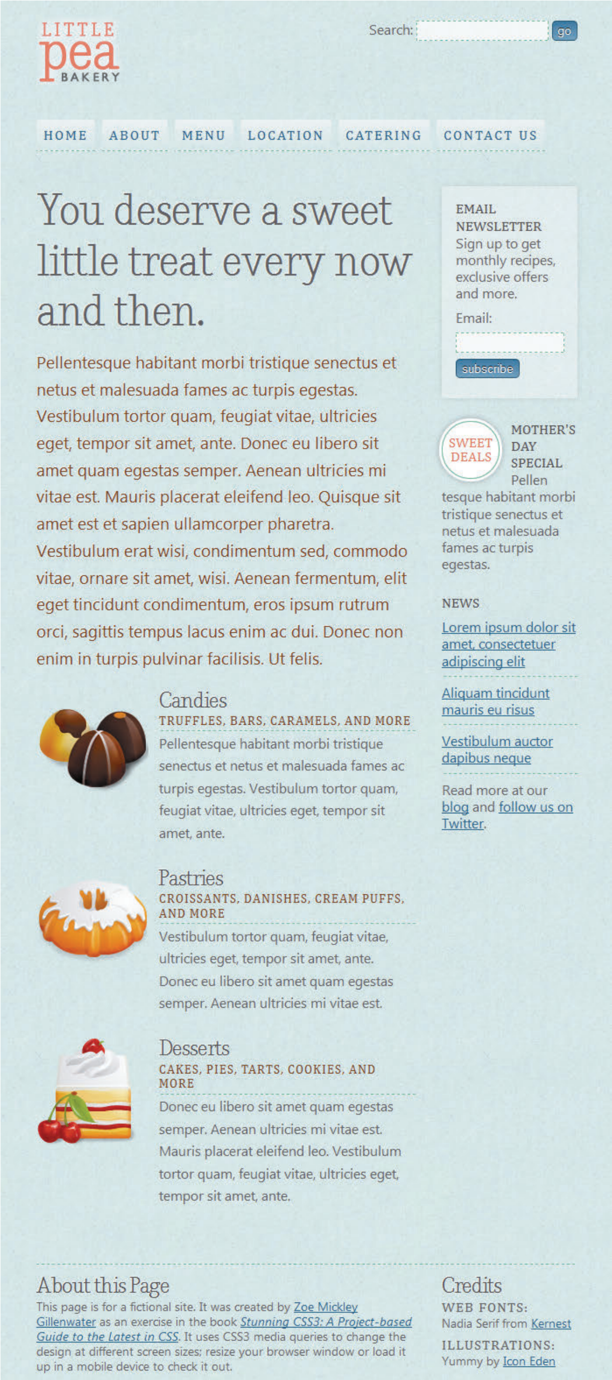


图 6-14　窄屏版的设计成品

6.5 针对移动设备来改变布局

如果你热衷于调整窗口大小来测试窄屏下的样式变化，你可能会注意到——即使样式已经按照我们的要求进行过调整，但在诸如移动设备之类的极窄宽度下，界面布局仍旧无法令人满意。那么，还剩下一件事需要我们完成：增加另一个媒体查询！

媒体查询是用来为移动设备简单快捷地定义样式的好方法。但是请注意我并没有说它是你实现移动版网站的唯一方法，你还可以为你的移动版网站增加服务端脚本或使用其他技术来改变其内容或功能。尽管媒体查询已经足以用来实现一个小型商务网站（比如本例中的面包坊网站）的移动版本，但面对一个大而复杂的移动版新闻网站，我们或许还需要使用其他一些额外的技术来显著地改变页面内容、导航和其他功能。而且，使用媒体查询来隐藏或切换大量不同的内容效率并不高——因为浏览器仍旧会下载它不需要的内容（请参阅 www.quirksmode.org/blog/archives/2010/08/combining_media.html）。因此，请不要把媒体查询当做解决所有移动版网站设计的救命稻草，它只是用来优化网站的移动版本的一项工具。

当你为移动版增加媒体查询时，目标是哪些屏幕尺寸呢？要知道手机屏幕的尺寸简直是千奇百怪，但绝大多数的流行手机——包括 iPhone 和诸多 Android 手机——其尺寸都是 320×480px，而其他的手机屏幕很少能大过这个尺寸。反观我们的面包坊页面，从 550px 就开始变开始出现错位，因此，将第三个媒体查询的目标定在 550 这个宽度，将适用于 320×480px 以及稍大于该分辨率的手机屏幕。

提示　如果你想针对某个特定的设备，可以到 http://cartoonized.net/cellphone-screen-resolution.php 查找其屏幕尺寸。

然而在正式开始增加这条媒体查询之前，让我们来聊聊有关设备宽度的话题。

6.5.1 什么是设备宽度

在媒体查询里有一个可用的媒体特性叫作 `device-width`，除此之外还有 `min-device-width` 和 `max-device-width`。设备宽度（Device width）即实际设备和显示器横向所能显示的像素数量，而不是可视区域的宽度。它意味着一台分辨率为 1280×800px 的桌面电脑，其设备宽度是 1280px。当你在媒体查询里指定 `device-width` 属性时，浏览器会忽略浏览器窗口的尺寸，而改为判断用户的屏幕分辨率。

一般来说手机没有“窗口”的概念，也可以理解为“窗口”的大小总是与屏幕尺寸相等，因此依据传统的方式理解“可视区域”其实并不合适，而“设备宽度”就显得更为合理。

但在这里也会遇到一些麻烦。Apple 公司并非始终将其产品的 `device-width` 设置为屏幕在横向上所能显示的像素数量，这一点与其他大多数手机都不同。尽管 iPhone 4 之前的机型以及 iPod

Touch 的屏幕尺寸都是 320×480 像素，它们的 Mobile Safari 会认定其设备宽度始终都是 320px，甚至当设备正运行于拥有 480px 宽度的横向模式也是如此。iPad 也有同样的命运，设备宽度始终是 768px，不管 iPad 的方向如何。而在 iPhone 4 下则更令人感到困惑，其分辨率高达 640×960，但设备宽度居然依旧保持为 320px。

这并不意味着你不能使用 `device-width`，但你应该本能地感觉到应该使用 `min-width` 和 `max-width`，而不是 `min-device-width` 和 `max-device-width`。它们在手机上的功能相同，但两组属性之间的主要差别在于是否会影响到非手机终端。举例来讲，一个 `max-width` 值为 550px 的媒体查询将同时影响设备宽度小于 551px 的手机和窗口窄于 551px 的桌面浏览器。但一个 `max-device-width` 值为 550px 的媒体查询则基本只会影响到移动设备，我想分辨率的宽度小于 551px 的桌面电脑恐怕并不多吧？所以，两组属性并无绝对的优劣之分，它们只是为你提供不同目标下的不同选择罢了。

6.5.2　第三次媒体查询

现在让我们正式开始针对 550px 宽的窗口加入第三次媒体查询：

```
@media screen and (max-width: 550px) {
}
```

请务必保证将这段代码置于第二段媒体查询（用以针对窄于 760px 的屏幕）的下方，因为它的代码同样会影响到移动设备，一个屏幕为 480px 像素宽的移动设备是符合“小于 760px”这个条件的。如果你将 550px 的媒体查询写到了 760px 这段的前面，那么后者的样式将会覆盖前者。这便是 CSS 级联的工作机制，后面的规则覆盖前面所声明过的同类规则。

实用的移动设备媒体查询

我们在移动设备的媒体查询中使用了 `max-width: 550px`，但还有各式各样的媒体查询可供你选择，这一切取决于你的目标设备或用户设置。这里列举有若干个示例，希望能够启发你写出适合自己项目需求的代码。

`(min-device-width:320px)and(max-device-width:480px)` 适用于 320×480px 分辨率的手机（如 iPhone 或 Android 设备），无论其是否处于横向或纵向模式。

`(max-width:320px)` 适用于处于纵向模式的 320×480 分辨率的手机。

`(min-width:321px)` 适用于处于横向模式的 320×480 分辨率的手机。

`(min-device-width:768px)and(max-device-width:1024px)` 适用于 iPad，与其所处的方向模式无关。

`(min-device-width: 481px) and (max-device-width: 1024px) and (orientation: landscape)` 仅适用于处于横向模式的 iPad。它也同样适用于宽大于高且分辨率小于 1024px 的桌面浏览器。

`(min-device-width: 481px) and (max-device-width: 1024px) and (orientation: portrait)`仅适用于处于纵向模式的 iPad。它也同样适用于高大于宽且分辨率小于 1024 的桌面浏览器。

如果你不希望两个媒体查询的条件发生重叠，可以将宽度最小值限定在 760px，比如：

```
@media screen and (min-width: 551px) and (max-width: 760px)
```

那么这条媒体查询将仅针对宽度在 551px 至 760px 之间的窗口，却不会影响宽度小于 551px 的移动设备。这件事可好可坏，它取决于项目的需求。但在我们的例子里，它意味着我们不得不在 550px 的媒体查询里重复书写许多与 760px 那条相同的样式规则，因为我们希望两种情况下大量样式保持一致。比如，我们希望引言段落在 760px 和 550px 两个版本的布局下都保持单栏显示。因此当这两条媒体查询发生覆盖时，我们仅需要在 760px 的媒体查询里定义单栏的样式，然后它会被继承到 550px 的版本里。

在我们的示例页面里，媒体查询之间的覆盖关系帮助我们重用了诸多样式并让 CSS 显得更简洁。然而在其他网站里，你可能希望不同宽度下的布局完全不同，这种情况下避免媒体查询之间的覆盖则更为合理。而且将它们分开只会带给你更少的烦恼，因为你不需要关心样式之间的继承关系。需要再次重申的是，对于这个问题并没有标准答案，它取决于你想要实现的具体需求。

在本例中，我们将保持 760px 的媒体查询代码不变，然后确保将 550px 的版本置于它的下方，这样宽度小于 551px 的窗口将同时应用上述两种样式。

1. 取消浮动

我们需要对移动版网站所作的主要改动便是取消浮动，这样整个页面才会保持为单栏。大多数移动版页面都是单栏的——仅仅是因为在如此小的屏幕下并没有多少空间可供栏目并排显示。

请将如下规则加入第三段媒体查询里：

```
@media screen and (max-width: 550px) {
   #content-main, #content-secondary {
      float: none;
      width: 100%;
   }
   #about, #credits {
      float: none;
      width: 100%;
   }
   #credits { margin-top: 1.6em; }
}
```

现在侧边栏改为在主内容栏的下方显示，而页脚部分的 Credits（贡献）区块则显示于 About（关于）区块之下（图 6-15）。贡献区块 `div` 所增加的上外边距将保持页脚里各个区块彼此分离。



图 6-15　在极窄的可视区域下，布局变成了单栏

2. 降低高度

另一项针对移动版页面的实用改动便是减少元素所占用的纵向空间，即减少用户在冗长的单栏页面中向下滚动的次数。

当使用移动设备近距离浏览时，宣传词和引言段落便无需使用这么大的文字，因此你可以通过增加 h1 和 h1 + p 的样式规则来减少字号：

```
h1 { font-size: 225%; }
h1 + p { font-size: 100%; }
```

图 6-16 展示出增加如上代码之后的变化。

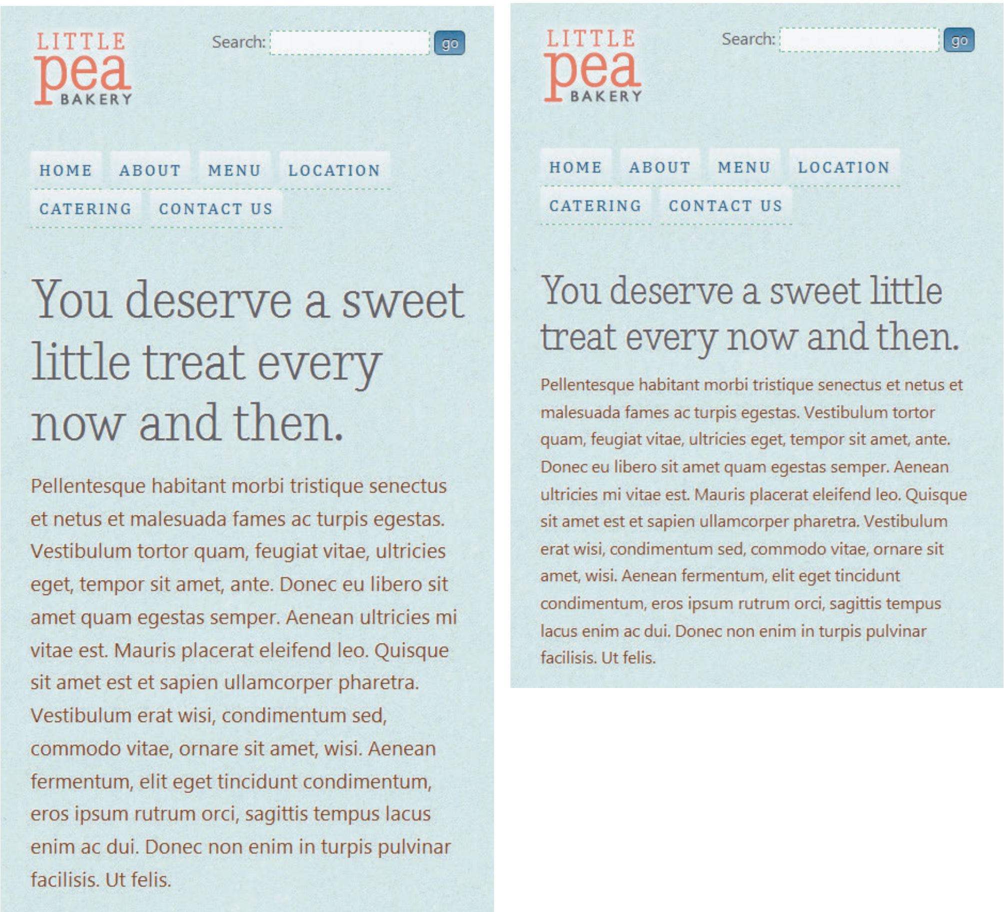


图 6-16　将引言段落的文字变小降低了用户在小屏移动设备上的滚动需求

顺着这条思路继续往下看，你会发觉在如此窄小的窗口下，产品图标却显得相当大，而它旁边的文字本可以占据更多空间的。不过幸运的是，我所选用的这套 Yummy 图片集包含了三种尺寸：128px、64px 和 48px。我们可以在媒体查询里将背景图片更换成 64px 的版本：

```
.feature { padding-left: 70px; }
#feature-candy { background-image:
                 url(images/icon_candy_64-trans.png); }
#feature-pastry { background-image:
                  url(images/icon_pastry_64-trans.png); }
#feature-dessert { background-image:
                   url(images/icon_dessert_64-trans.png); }
```

提示　除了换成不同的图片之外，你还可以使用 background-size 属性来调整背景图片的尺寸。但这个方式的缺点是会造成图片有少量锯齿，但优点则在于只需载入一套图片。

至此，特色产品区域占用了更少的高度而且文字和图片看起来更平衡了（见图 6-17）。

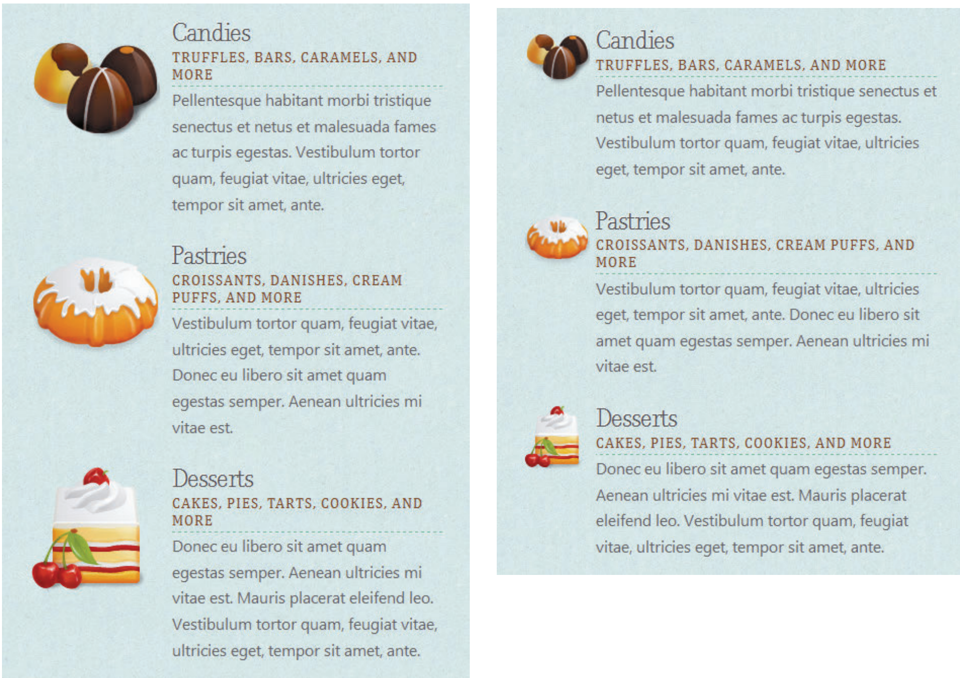


图 6-17　在极窄情形下减少特色产品区域的图片尺寸使它们相对于文字看起来更为平衡了

接下来，让我们看看邮件订阅区块。其中的文本域占据了它的整个宽度，使得同一行里的标记文字和按钮甚至在稍大一些的手机屏幕里都没有了立足之地。因此需要为其增加如下的样式规则：

```
#form-newsletter * { display: inline; }
#form-newsletter input[type=text] { width: auto; }
```

这些修改让邮件订阅区块显得更紧凑了（见图 6-18）。在纵向模式的手机屏幕里，订阅按钮将会挤到第二行，但即使如此整个表单总体上仍然很好地利用了空间。

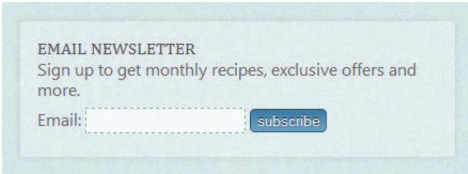


图 6-18　现在邮件订阅区块里的表单元素在手机屏幕的横向模式下全部显示于同一行了

最终，我们再对页脚进行一些细小的修改来降低其高度。由于还有足够的空间可将标题（如 Web Fonts）显示在详情（如 Nadia Serif from Kernest）的一旁，所以让我们将贡献区块 `div` 里的 `dt` 元素设置为浮动：

```
#credits dt {
   clear: left;
   float: left;
   margin: -.05em .2em 0 0;
}
```



图 6-19　标题和详情被调整到同一行之后，页脚的贡献区块所占用的位置变少了

3. 防止标题元素发生重叠

在窄小的移动屏幕里，页面元素彼此之间发生重叠的可能性无疑会增加。在我们的示例文件中，标题便存在这个问题。在 550px 宽的可视区域下，搜索表单可以正常地显示于 logo 一旁，但在大概 400px 宽时它们便开始发生重叠。如果用户设置了更大的字体，那么重叠现象将出现得更早。

为了降低发生重叠的概率，我们在第三个媒体查询里增加如下的规则来减少搜索表单里文本域的宽度：

```
#form-search input[type=text] { width: 100px; }
```

接下来，在 550px 的媒体查询下面新增第四条。这条媒体查询针对的是宽度少于 401px 的窗口：

```
@media screen and (max-width: 400px) {
}
```

在该媒体查询里增加一条新规则，作用是使搜索表单里的标签显示成块元素，这样它将单独占据文本域之上的一行：

```
@media screen and (max-width: 400px) {
   #form-search label { display: block; }
}
```

至此，搜索表单在 550px 和 400px 宽的可视区域之下所占用的空间更少了，甚至在 320px 宽的手机屏幕里都不会与 logo 发生重叠（图 6-20）。

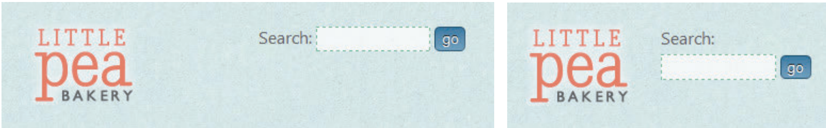


图 6-20　搜索表单现在同样占据了更少的空间，而且在手机屏幕上与 logo 发生重叠的可能性更小了（左 480px，右 320px）

6.5.3　针对高清显示屏来改进界面

iPhone 4 有一种新型的屏幕叫作“视网膜屏”（Retina Display），它比旧版的 iPhone 和 iPod Touch 的屏幕分辨率要高。其分辨率是 640×960，但它所展示的区域却与旧版 iPhone 相同，这是因为它将两个硬件上的像素看作是一个 CSS 的像素。这就意味着你在 CSS 里所定义的每个像素都由两个实际的像素来显示——这就是它高清的原因，也是缘何其设备宽度仍旧为 320（640 的一半）。

大多数情况下，你会希望所有版本的 iPhone 看到的样式都相同，但你也可能希望利用 iPhone 4 视网膜屏幕的优势来显示更高清的图片。比如说，在视网膜屏上我们的产品图片与清晰的文字相比，看起来有一些像素化。

为了针对 iPhone 4 设置样式，你可以使用一个名叫-webkit-min-device-pixel-ratio 的媒体特性属性，并将其值设置为 2：

```
@media screen and (-webkit-min-device-pixel-ratio: 2) {
}
```

它让媒体查询仅当设备像素与 CSS 像素之比为 2∶1 时才生效，目前来说仅有 iPhone 4 才符合条件，但其他 Apple 设备在未来也可能装备上视网膜屏。因此为了使这条媒体查询更经得起时间的考验，我们给它的媒体特性增加另一个条件来使其仅仅针对 iPhone 4 的小屏：

```
@media screen and (-webkit-min-device-pixel-ratio: 2)
¬ and (max-width: 480px) {
}
```

说明　还有一种分辨率的媒体特性可以让你根据设备的 dpi 最小值来设置样式，然而目前 Webkit 浏览器并不支持该特性——其中也包括 iPhone 4 所使用的 Mobile Safari。其他高分辨率的显示设备可能支持它，所以在你的目标设备上测试一下，看它是否有用。

现在我们可以开始为 iPhone 4 提供更大的图片了，然后用 background-size 属性来将其缩小，这样可以在同样的空间有效地显示更多的像素信息。请在新的媒体查询里增加如下的规则：

```
.feature {
    -webkit-background-size: 64px 64px;
    background-size: 64px 64px;
}
```

```
#feature-candy { background-image:
                 url(images/icon_candy_128-trans.png); }
#feature-pastry { background-image:
                  url(images/icon_pastry_128-trans.png); }
#feature-dessert { background-image:
                   url(images/icon_dessert_128-trans.png); }
```

这些图片的实际尺寸（128×128）比我们希望它们所展示的尺寸（64×64）整整大了一倍。当我们使用 `background-size` 将其缩小至 64px 来显示时，一个普通的浏览器便有了两倍数量的像素显示一个 64px 的图片。因此当 iPhone 4 需要将每个像素双倍显示时，它就已经有了两个像素可供显示，这样就不会遇到将一个像素放大两倍却造成模糊的情况了。图片将看起来更加清晰。

这仅仅是我们针对 iPhone 4 优化的其中一步。如果你想了解更多关于利用其高清显示屏的想法，请参阅 Luke Wroblewski 的“Designing for the Retina Display (326ppi)”一文（www.lukew.com/ff/entry.asp?1142）。

6.5.4 可视区域的meta标签

如果你现在保存并用桌面浏览器测试该页面，当你将窗口缩小到一定宽度它会如你所想地展现出改进结果；但当你使用诸如 iPhone 或 Android 之类的智能手机来测试时，你可能会惊奇地发现：所有的媒体查询都不生效——页面仍展示为普通的样式，即一个全局缩放后的两栏布局（见图 6-21）。

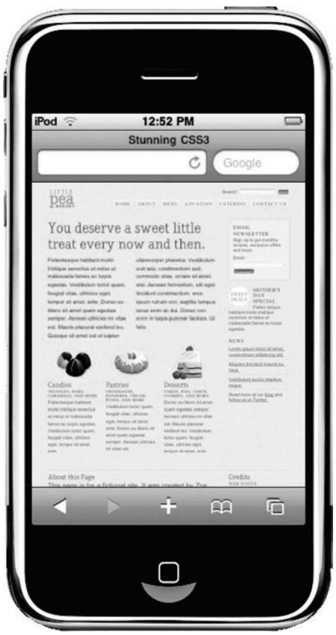


图 6-21　在很多手机设备上，页面展现为一个两栏的布局，并未应用我们在第三个媒体查询里所设置的样式

这是因为许多智能手机都使用一个比实际屏幕尺寸大很多的虚拟可视区域，这样做的目的是为了避免那些未针对手机进行优化的页面在 320px 的实际可视区域下被挤压得变形。移动 Web 开发专家 Peter-Paul Koch 把虚拟的可视区域称作“布局可视区域”（Layout Viewport），而实际的可视区域则叫作“视觉可视区域”（Visual Viewport）。

当你在手机上使用布局可视区域来浏览页面时，手机浏览器将缩小到最高级别，这样一来整个布局可视区域刚好可以符合屏幕的尺寸。这会使所有东西看起来很小，但它能保证界面布局看起来与使用一个典型的桌面浏览器访问时看到的一样。不同的手机浏览器使用不同的界面可视区域的宽度值——iPhone 和 iPod Touch 的 Mobile Safari 使用 980px，Android Webkit 使用 800px，Opera 使用 850px——这里的重点是当你希望手机坦白地告诉你其屏幕的分辨率时，它们都假装自己的屏幕比实际所拥有的屏幕要大。

说明　如果你想详尽地了解手机的可视区域及其 `meta` 标签，请参阅 Peter-Paul Koch 的文章——“A tale of two viewports—part two”（www.quirksmode.org/mobile/viewports2.html）。

幸运的是有个特殊的 `meta` 标签，其唯一的目的便是告诉手机浏览器你的网站针对它们进行了优化，并且可以让你自定义界面可视区域的尺寸和缩放级别。

1. 它如何工作

这个为手机而生的标签被称为“可视区域 `meta` 标签”（Viewport meta Tag），使用时你需要将其 `name` 属性设置为 `viewport`。它写起来形如：

```
<meta name="viewport" content="">
```

在 `content` 属性内，你可以填入任何用于处理可视区域的指令。表 6-3 罗列了所有可以填入 `content` 属性的指令。

表6-3　可视区域meta标签的content属性可用值

属　　性	描　　述
width	可视区域的宽度，单位是像素。其取值可以是一个实际的数字或关键词device-width
height	可视区域的高度，单位是像素。其取值可以是一个实际的数字或关键词device-height
initial-scale	某个页面首次被显示时可视区域的缩放级别。取值为1.0将使页面以其实际尺寸显示，无任何缩放
minimun-scale	可视区域的最小缩放级别。它表示用户可以将页面缩小的程度。取值为1.0将禁止用户缩小至实际尺寸以下
maximun-scale	可视区域的最大缩放级别。它表示用户可以将页面放大的程度。取值为1.0将禁止用户放大至实际尺寸以上
user-scalable	指定用户是否可以对页面进行缩放。设置为yes将允许缩放，no则禁止缩放

可视区域 `meta` 标签由 Apple 发明且尚未成为标准。然而，除 iPhone 之外的很多手机浏览器都支持该标签。

2. 添加到页面中

现在让我们把可视区域 meta 标签添加到面包坊页面中。请在页面的 head 中加入如下代码：

```
<meta name="viewport" content="width=device-width">
```

这将告诉手机浏览器你想将布局可视区域的尺寸设置为设备宽度，即屏幕的尺寸。现在当你使用诸如 iPhone 或类似的设备查看该页面，你将发现手机浏览器正以 320px 的宽度显示布局，证明我们的媒体查询代码生效了（见图 6-22）。

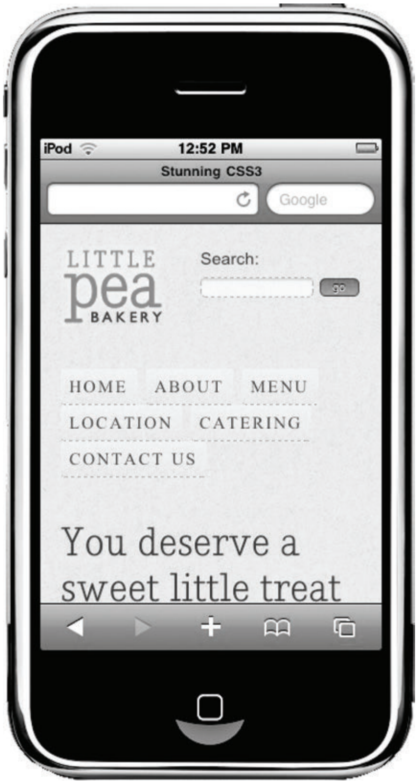


图 6-22　增加了可视区域 meta 标签之后，iPhone 便以 320px 的宽度来显示该页面，不会扩大至 980px 了

测试媒体查询

当你使用媒体查询进行开发时，会发现大量的时间花费在来来回回地缩放浏览器窗口以确认样式是否符合自己的预期上。请允许我向你介绍一些经验帮助你更快捷、顺利地测试媒体查询：

首先，一个名叫 ProtoFluid 的 Web 应用（http://protofluid.com）是专门为测试媒体查询而生的。填入一个 URL 然后选择你想要测试的设备（如 iPhone 或 Motorola Droid）的屏幕尺寸，它会弹出一个窗口并强制它符合你所要求的尺寸，还提供给你一个按钮以便你在横向和纵向模式之间来回切换。但是需要注意的是它并不会真正模拟这些设备，仅是建立了一个与其屏幕尺寸相同的窗口而已。比如说你所看到的内容并不会准确符合可视区域 `meta` 标签的设置。而且，ProtoFluid 也并不支持 `device-width`，如需测试，请改用 `min-width` 和 `max-width`，接着测试完成后再换回 `min-device-width` 或 `max-device-width`。

另一个我惯用的测试方法是使用 Firefox 的 Web Developer 扩展（http://chrispederick.com/work/web-developer）为窗口设置一个与屏幕大小不同的可视区域尺寸。展开 Resize 目录，你可以增加任意多的窗口或可视区域尺寸。你可以增加一些符合常用设备屏幕尺寸的配置，比如纵向模式下 iPhone 屏幕的可视区域是 320×356，横向模式下则是 480×208，还有那些被你作为条件写入样式文件的媒体查询里的尺寸。然后就可以通过对目录菜单里一次简单的点击来将浏览器缩放至预设的尺寸并预览效果了。再次提醒：这种方法仍旧不会真正模拟手机的行为，仅能方便你测试不同的窗口宽度而已。

然而，如果你将 iPhone 转向到横向模式，页面仍会显示为 320px 宽，而非 480px。Mobile Safari 直接对页面进行了放大，而不是真正去改变界面可视区域的尺寸。这样做的结果是 logo 和其他图片都变模糊了（见图 6-23）。



图 6-23 在横向模式下，iPhone 仍旧以 320px 的宽度进行显示，却将界面放大至 480px

还记得吗？出现这个问题是由于无论在横向还是纵向模式，iPhone 和 iPod Touch 所返回的设备宽度都是 320px。为了让 Mobile Safari 在横向模式下以 480px 的宽度显示布局可视区域，你需要禁止它为了符合窗口尺寸便将界面放大至 320px。可以把 `maximum-scale` 设置为 `1.0` 来禁止浏览器（和用户）放大界面至实际尺寸以上，并将其写入 `meta` 标签：

```
<meta name="viewport" content="width=device-width,
¬ maximum-scale=1.0">
```

说明 iPad 同样会识别且应用可视区域 meta 标签，尽管其屏幕尺寸更像桌面显示器而非手机。不过幸运的是，我们所写的 meta 标签并未对 iPad 产生影响，其界面尺寸仍旧是 768px（纵向）和 1024px（横向），正如设想的那样。

现在，当我们使用横向模式浏览该页面时，Mobile Safari 将保持缩放级别在 100%，即强制布局可视区域放大至 480px 以符合屏幕尺寸（见图 6-24）。

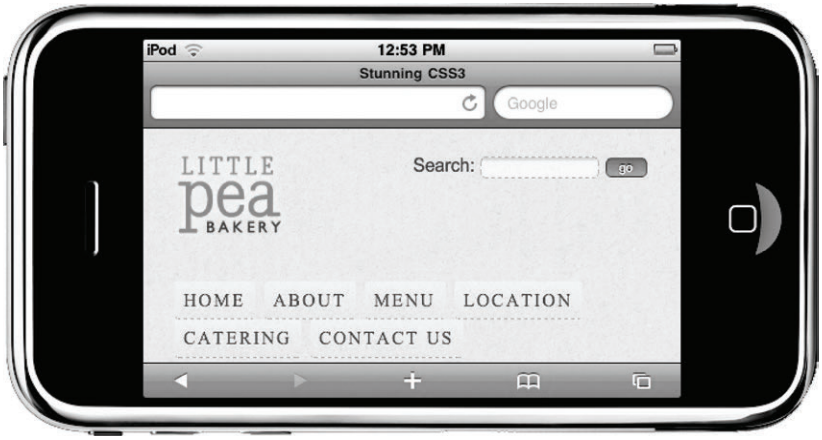


图 6-24 改变可视区域 meta 标签的 maximum-scale 属性，强制 iPhone 将横向模式下的布局放大至 480px

6.6 针对其他浏览器的兼容方案

不支持媒体查询特性的浏览器（比如 IE 8 及之前的版本和 Firefox 3.1 及之前的版本）访问时仍将显示普通版本的页面。尽管在极窄或极宽状态下它看起来并不佳，但绝大多数的用户并不会使用这些尺寸访问。你可以在页面里增加 min-width 和 max-width 来保证这些浏览器避开这些极限情形；不过请记得在媒体查询里对不需要这些设置的浏览器加入样式覆写。

当然，这个并不会改善移动设备的访问体验。幸好大多数流行的手机都支持媒体查询（www.quirksmode.org/mobile/browsers.html）。但是，大量不支持媒体查询的手机也支持 handheld 媒体类型，因此你可以为这些浏览器增加专门为其定制的样式文件。Dominique Hazaël-Massieux 撰写的“Return of the Mobile Stylesheet”一文（www.alistapart.com/articles/return-of-the-mobile-stylesheet）介绍了一个方法，它可以根据手机浏览器是否支持 handheld 媒体类型或媒体查询来

提供不同的样式文件。

如果这些解决方法仍旧无法适用于你的项目，还可以使用 JavaScript。Wouter van der Graaf 所编写的 css3-mediaqueries-js（http://code.google.com/p/css3-mediaqueries-js）便是其中的佼佼者。你所要做的只是将脚本引入到页面里——它将自动分析存在于 CSS 中的媒体查询代码并使其在版本陈旧的浏览器上工作。

你可以在下载的本章的示例文件中找到该脚本，并将它引入页面的 head 中：

```
<script src="scripts/css3-mediaqueries.js"></script>
```

说明　本书编撰之时，示例文件中所包含的是该脚本的最新版本（发布日期是 2010 年 3 月）。但当你阅读时它可能已经有了最新的版本，请你下载并予以替换。可以到 http://code.google.com/p/css3-mediaqueries-js 查看版本。

保存页面并在原生并不支持媒体查询特性的浏览器（如 IE 8）中进行查看。缩放窗口你便会看到界面布局开始根据媒体查询的设置进行变化了。简单如斯，夫复何求！

对用户来说，使用该脚本唯一的缺点便是会增加一次额外的 HTTP 请求，并需要因此下载 16k 的数据。正因如此，你可能希望仅当浏览器切实需要它时才真正引入它。针对 IE 6-8，我们可以使用条件注释：

```
<!--[if lte IE 8]>
<script src="scripts/css3-mediaqueries.js"></script>
<![endif]-->
```

尽管诸如 Firefox 3.1 之类的浏览器不支持条件注释，同时也不支持媒体查询，但其实这些浏览器的用户并不多。对于本例来说，与其强迫所有浏览器都下载这个额外的脚本，倒不如让一些不支持媒体查询特性的非 IE 用户错过这个功能。

说明　如果你正在使用 jQuery，那么这里有两个插件可以使媒体查询实现兼容：www.protofunc.com/scripts/jquery/mediaqueries 和 www.csslab.cl/2009/07/22/jquery-browsersizr。

6.7　大功告成

至此，面包坊页面针对若干不同的窗口尺寸都有了其独特的布局，使得它在桌面宽屏显示器和窄屏的移动设备上既美观又易于使用（图 6-25）。

说明　完成版页面所展示的效果代码都在 media-queries_final.html 页面里，它在你下载的本章的示例文件之中。



图 6-25　面包坊页面在不同的尺寸下有着不同的设计（另见彩插图 6-25）

第 7 章

布　局　秀

尽管一直以来总是鼓励使用 CSS 来控制页面布局，但 CSS2.1 的工作机制中可以用于布局的部分并不多。开发者通常都不愿意使用绝对定位，因为那太不灵活；浮动定位被广泛运用，但对于全页多列布局来说，它总存在这样那样的小毛病，功能上也有很多限制。CSS3 引入了许多的布局机制，能够让构建一个多列布局变得更加轻松，同时，在 CSS2.1 规则下比较难实现的一些复杂布局表现，如今也变得更加容易。虽然这些特性还只是实验性的，这一章中你还是可以了解到未来的布局技术发展趋势，同时也可以学习到如何立刻使用新的弹性布局模型的实战技巧。

你将学到

我们将在一个页面中使用下面这些 CSS3 技术来创建多列布局，并放置一些小部件在页面上：

- 弹性布局模型；
- box-sizing 属性。

两种 CSS3 中即将实现的布局在本章也会提及，即模板布局和网格定位。

7.1　重大改变

在本书中，只有那些能够通过某种渐进增强的方法实际运用到工作当中的 CSS3 技术才会涉及。但在这最后一章中，将会涉及一些浏览器支持度不佳，甚至没有被任何浏览器支持的 CSS3 属性，诚然，浏览器的支持对于页面的意义远远比那些旧式浏览器可以完全忽略的视觉效果重要。但是，了解它们还是很有必要的，因为这些属性所提供的功能，将会使 Web 设计和开发的过程发生巨大的改变。而且还是有些方法可以实际使用这些属性，同时又保证在不支持的浏览器中有较好的降级体验的。

因此最后这一章主要会描述几年以后构建 Web 站点的方式。了解了这些新技术以后，你可以通过一些有限的方法在网站中逐渐开始使用它们，那么当这些技术得到广泛的运用和支持的时候，你就已经站在业界的最前沿了。

目前最完善，支持度最好，实际可以使用的新布局技术通常称为弹性布局模型（本章将会主要描述这种布局技术）。弹性布局模型能指定布局元素的排列方式是水平还是垂直的，这些布局元素彼此对齐的方式，以及这些元素如何使用可用空间。这听起来没有什么大不了的——CSS2.1 里

面不是早已经有属性可以做到这些了吗——但是 CSS2.1 做不到像弹性布局模型这样轻松和灵活。

说明　弹性布局模型并未替换 CSS 2.1 的盒模型，它们可以同时使用。

7.2　无需浮动或绝对定位，创建多列布局

弹性布局模型引入了一个专用体系来创建多栏或者多列的布局，这跟浮动或者绝对定位截然不同。一个布局示例就能很好地说明弹性布局模型的工作机制，请在 www.stunningcss3.com 下载本章的练习文件，然后在代码编辑器中打开 flex-box_start.html 文件。

这个页面和第 6 章的原型页面是一致的，但媒体查询的部分被移除了，主要是想让例子更简单一点，而且原本使得各栏依次排列的 CSS 布局代码也被去掉了，这样所有的 div 元素都自上而下垂直排列，因为所有的块级元素默认都是这样排列的（图 7-1）。通过使用弹性布局模型，这些块级元素将会再次水平排列。



图 7-1　不使用 float 时，div 依次垂直排列

为了让这些区块能够水平排列，首先要将它们的父容器的 display 属性设置成 box，这是 CSS3 为著名的历史悠久的 CSS 属性 display 设置的一个新值。在这里一个命名为 content 的 div 是两个主要内容区块（一个是 content-main，一个是 content-secondary）的父容器，在第 6 章中页面上并没有这个父容器，之所以添加它，是因为这是弹性布局模型的需要。加入了这个容器之后，在页面 head 区样式表内容里面增加如下的一条关于#content 的规则：

```
#content {
   display: -moz-box;
   display: -o-box;
   display: -webkit-box;
   display: box;
}
```

将 display 设置成 box 会导致 div 成为 W3C 术语中所定义的*弹性盒元素*，通常简称为*盒元素*，这会告知浏览器，此 div 和它的子节点都将使用弹性盒模型来布局。Firefox 和 Webkit 核心的浏览器——目前实现了对弹性盒模型支持的浏览器——可以通过-moz-box 和-webkit-box 属性来应用这一特性。对于无浏览器前缀的 box 值，目前还没有浏览器可以支持，Opera 还不支持 -o-box，添加的目的仅仅是为了向后兼容。

额外的包裹容器

为了实现布局，上面的例子在 HTML 代码里面添加了一个额外包裹容器来容纳名为 content-main 和 content-secondary 的 div 容器，这展示了弹性布局模型的一个缺点：相对浮动布局，它需要增加额外的无实际意义的嵌套 div 容器。如果要使得嵌套在内的 div 容器水平排列，外层的 div 包裹容器必须设置 display:box 属性。相对弹性布局模型带来的好处而言，少量的包裹容器并不是什么大问题，当然，知道这个细节，还是非常有意义的。

IE 9 兼容性

在本书撰写的阶段，如果使用-ms-前缀的话，IE 9 平台预览版是可以支持弹性布局模型的，但是当前的 IE 9 beta——它比平台预览版要新——却反而又不支持了。很明显微软在发布 beta 版时决定去掉这个功能。IE 9 的最终版是否会重新支持这个特性，是否还需要使用浏览器前缀，这些问题尚无定论。或许当你阅读本书时可以尝试测试一下 IE 9 正式版的兼容性，当然为安全起见，不如直接把所有的浏览器前缀都加上。

如果想让子元素按照水平排列，就应该进一步使用 box-orient 属性，包括各种浏览器前缀的。

```
#content {
   display: -moz-box;
   display: -o-box;
   display: -webkit-box;
   display: box;
   -moz-box-orient: horizontal;
   -o-box-orient: horizontal;
```

```
    -webkit-box-orient: horizontal;
    box-orient: horizontal;
}
```

如果容器的 display 属性设置成 box 的话，浏览器会将容器的 box-orient 属性自动设置成为 inline-axis，这在英语里面的意思就是水平排列，跟 horizontal 这个值的含义是一样的：这会让容器的子元素水平地并排排列，而不是从上至下。所以，实际上这里根本无需再设置 box-orient 属性，子元素会自动水平排列。这里的设置只是为了更清晰地表明如何使用这个属性。

此条规则能让页面的外观发生奇妙的变化，content-main 和 content-secondary 这两个 div 现在比邻而居，但是如果发生了意外呢？比如这两个 div 的宽度不断增长到一个足够的值时，页面上将无法完整显示这两个 div，这个时候，浏览器会显示一个水平的滚动条（见图 7-2）。

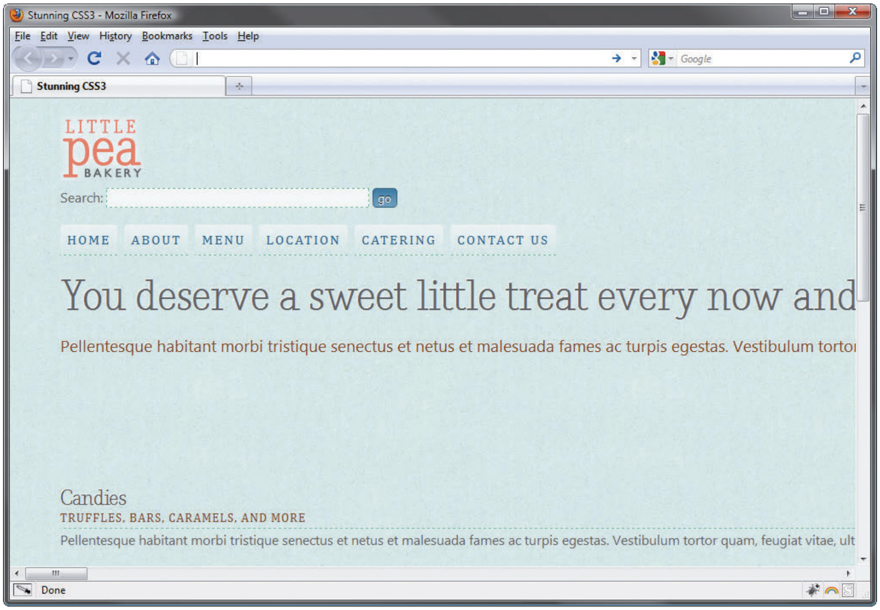


图 7-2　两个内容 div 水平排列，但是由于宽度太宽而无法完整显示

造成这个现象的原因是弹性容器的子元素会增减自身的宽度来适应内部的内容，这被称为固有大小，跟没有指定宽度的浮动元素所表现的现象是一致的。div 中最长的文本或者最宽的图片决定了 div 的宽度。很多时候这恰好是满足期望的，但在这个例子中则不然。为了解决这个问题，可以给每个 div 指定 width 或者 max-width 属性，也可以使用 box-flex 属性让其中一个或者是全部 div 具有自适应的宽度。

7.2.1　子元素自适应

box-flex 属性实际上并不能阻止子元素保持固有大小，但是它会强制子元素内部的内容自

动适应而不会把子元素本身撑大。这就让这些子元素能够弹性适应父容器的宽度。如果子元素的固有宽度之和小于父容器的宽度，子元素会自动增加宽度以填满父容器，当相反的情况出现时，子元素会减小宽度。

具体增加或者减小的量取决于子元素们的固有宽度的比例，而不是某个绝对的数值。弹性适应的元素是按照特定比例来弹性适应的。举例来说，一个 box-flex 值为 2 的元素的宽度将会两倍于一个 box-flex 值为 1 的元素，额外的空间会按照 2∶1 的比例分配。

实际的例子能很好地说明这一切。图 7-3 展示了一个灰色的 800 像素宽的 div，内置了两个保持固有大小的子元素，子元素的宽度由内部的文本决定，其中黄色的那个是 99 像素宽，粉色的那个是 493 像素宽。父容器的内部还有 208 像素的宽度是多余的。如果把两个子元素的 box-flex 属性的值都设置成 1 的话，它们就会按照 1∶1 的比例瓜分剩余的 208 像素宽的空间，换句话来说，它们平分了剩余的空间（图 7-4）。如果将粉色 div 的 box-flex 设置成 2 的话，它分到的剩余空间就是黄色 div 的两倍（图 7-5）。

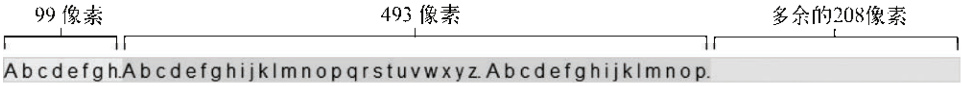


图 7-3 黄色和粉色的 div 的宽度由内部的内容决定，这样灰色的 div 内部还留有额外的宽度（另见彩插图 7-3）

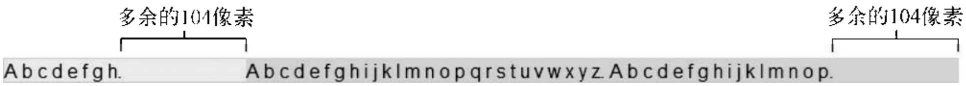


图 7-4 两个 div 各增加了 104 像素的宽度（另见彩插图 7-4）

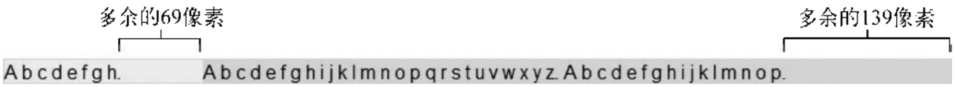


图 7-5 由于 box-flex 的比例是 2∶1，所以黄色 div 增加了 69 像素，粉色增加了 139 像素（另见彩插图 7-5）

如果子元素的总宽度要比父容器大时，box-flex 的工作原理是一样的——超过的部分会按照 box-flex 值设置的比例减去。举例来说，在图 7-6 和图 7-7 中，父容器的宽度被设置成为 500 像素，这就比子元素的总宽度要小。在图 7-6 中，子元素的 box-flex 的值都被设置成 1，因此它们的宽度同时减小 46 像素。在图 7-7 中，粉色的 div 的 box-flex 的值设置为 2，因此它减小的宽度是黄色的两倍。

说明 区块因 box-flex 而被拉伸或缩小的程度取决于它的最大和最小宽度，可以直接指定也可以继承得来，所以有时候最后的宽度值可能并非如你所想。



图 7-6　为了适应 500 像素的父容器，所有子元素都减小了 46 像素

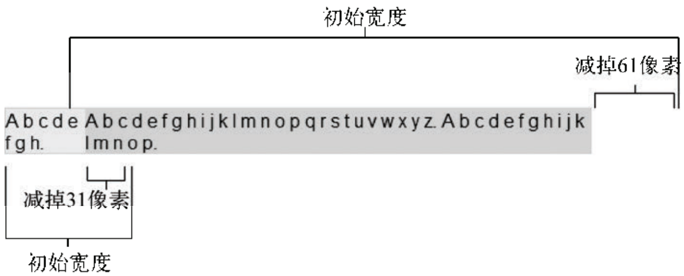


图 7-7　由于比例是 2∶1，所以粉色的 div 减小了 61 像素，黄色的减小了 31 像素

在本书的例子里面，会通过把 box-flex 的值都设置成 1 从而让 content-main 和 content-secondary 两个 div 的宽度均等地自适应：

```
#content-main {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
    margin-bottom: 40px;
}
#content-secondary {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
    margin-bottom: 40px;
}
```

现在两个子元素都能和父容器保持和谐了（图 7-8）。但是 Firefox 和基于 Webkit 的浏览器在决定每个子元素的宽度时有所不同：图 7-8 是 Firefox 中的效果，图 7-9 是 Chrome 中的效果，它们的父容器的宽度都是一样的。没法说哪个浏览器是正确的，也许技术上它们都错了，因为 W3C 在关于浏览器应该如何检测元素的固有大小这个问题上并没有提供足够的细节描述。

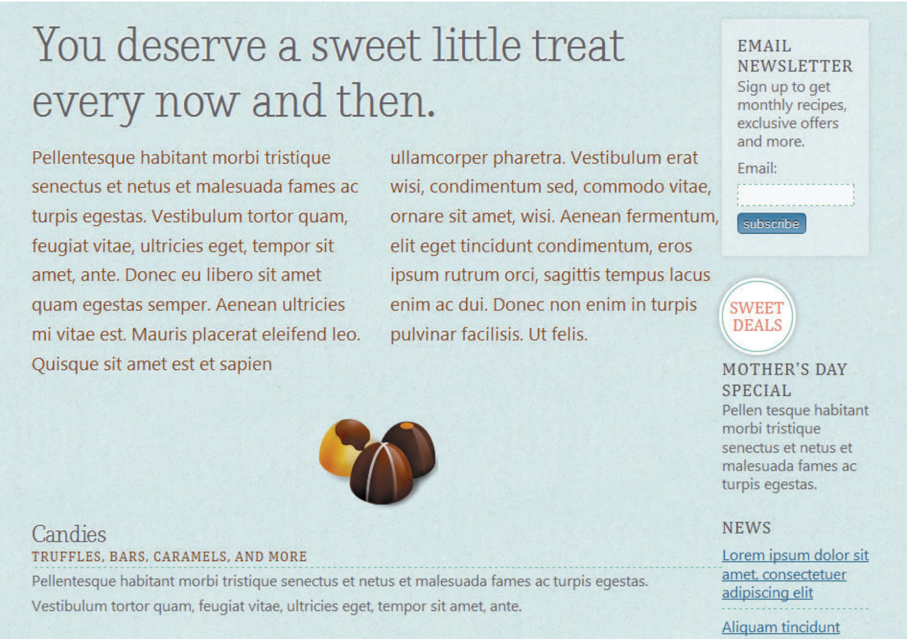


图 7-8　两个 div 在 Firefox 下的表现

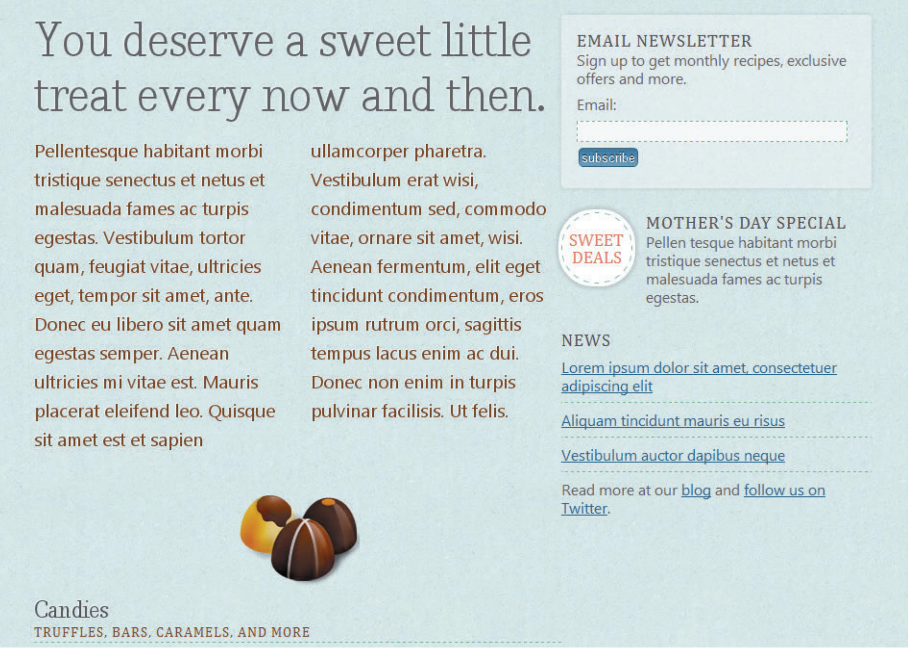


图 7-9　两个 div 在 Chome 下的表现，请注意边栏相对于 Firefox 变宽了

说明 如果盒子是纵向层叠而不是横向，那么 box-flex 值将影响其高度而不是其宽度。它所控制的弹性空间与盒子布局的方向一致。

正因为如此，很明显在这个例子里把两个 div 都设置成自适应是无法满足设计需求的。相反地，应该给边栏一个特定的宽度，以及为它设置一个左边距来让它和主要内容之间保持一定间距。

```
#content-secondary {
    width: 16em;
    margin: 0 0 40px 40px;
}
```

此时边栏的宽度将会固定在 16em，而主要内容将会自适应扣除了边栏和边栏的边距之后的宽度（图 7-10）。即便总的布局模式不是自适应的流式布局，这个设定也是有效的，举例来说，如果父容器被设置成为 960 像素定宽，边栏会占据 16em，主要内容区域将会延展到扣除边栏宽度的剩下宽度值。box-flex 属性的作用是让子元素自适应父容器的宽度，而不是自适应可视区域的宽度，即使父容器是定宽的也如此。

说明 至此为止的所有修改都在 flex-box_1.html 页面里，它在你下载的本章的示例文件中。

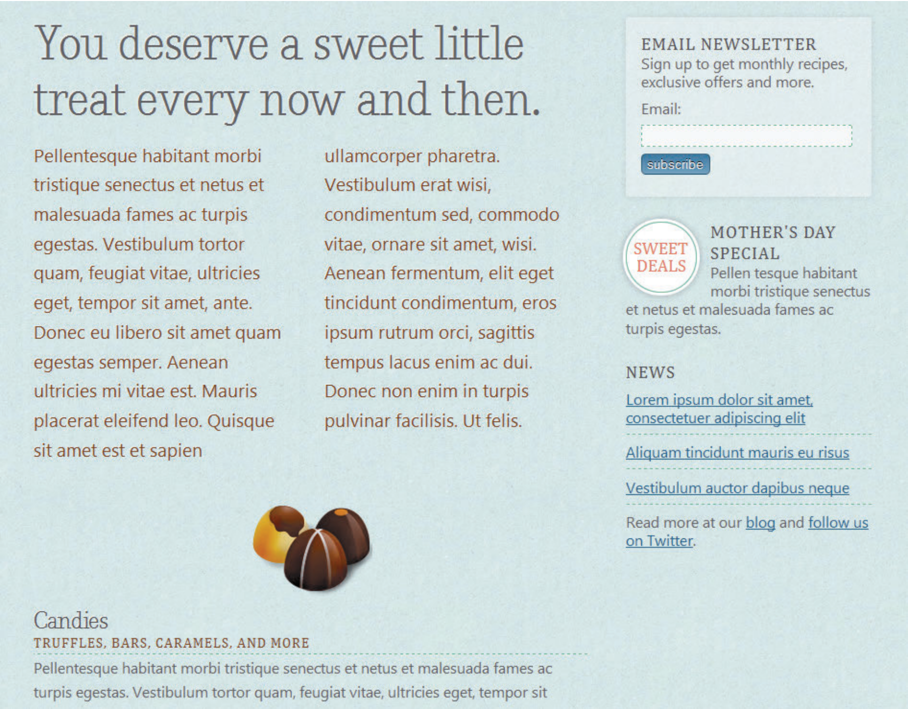


图 7-10　边栏 16em 宽，剩余的部分都由主要内容区域来填充

7.2.2 增加栏目

这个布局例子表明了弹性布局模型的一个优点：可以轻易地把不同的元素按照不同的数值单位水平排列。主要内容区域的宽度实际上是按照百分比指定的，边距是以像素指定的，边栏则是以 em 指定的。不使用弹性布局模型的话，这种混合模式也是可以实现的，但是会更困难、更杂乱。

相对地，使用弹性布局模型来创建一个混合两栏布局是很容易的，下面是一个小结：

(1) 将栏目的父容器的 display 属性设置成 box；

(2) 将父容器的 box-orient 属性设置成 horizontal；

(3) 将边栏设定一个固定宽度；

(4) 使用 box-flex 特性来让主要内容栏填充剩余的父容器空间。

如果希望加入更多的栏目，那么可以为新增的栏目指定一个特定的宽度，也可以使用 box-flex 属性来让它们自适应宽度。你无需为新增的栏目来重新调整所有的栏目的宽度——所有的东西都会自动调整。

用个例子来说明一下，如果在底部新增一个栏目，那么首先需要像内容区域那样，为底部已经存在的两个 div 设置弹性布局。首先在#footer 规则中添加 display 和 box-orient 属性：

```
#footer {
   display: -moz-box;
   display: -o-box;
   display: -webkit-box;
   display: box;
   -moz-box-orient: horizontal;
   -o-box-orient: horizontal;
   -webkit-box-orient: horizontal;
   box-orient: horizontal;
   padding: 10px 0;
   border-top: 1px dashed #3C9;
}
```

然后为底部内已经存在的两个 div 增加如下规则：

```
#about {
   -moz-box-flex: 1;
   -o-box-flex: 1;
   -webkit-box-flex: 1;
   box-flex: 1;
}
#credits {
   width: 10em;
   margin: 0 0 40px 40px;
}
```

通过这些修改，底部的内容已经形成了两栏布局。接下来再将它改变成为三栏布局。首先在 about 和 credits 这两个 div 中增加第三个 div。从 flex-box_2.html 中复制以下的 HTML 并粘贴到练习文件当中：

```
<div id="learn-more">
    <h2>Learn More</h2>
```

```
    <p><a href="http://www.w3.org/TR/css3-mediaqueries/"
    >Media queries</a> are a way of tailoring the site
    design to the characteristics of each user's display,
    using regular CSS embedded in your main style sheet.
    </p>
</div>
```

然后再为这个 learn-more 容器增加一条新的规则来设置它的宽度和边距：

```
#learn-more {
    width: 10em;
    margin: 0 0 40px 40px;
}
```

在无须对其他 CSS 做任何改动的情况下，底部的两栏布局已经变成三栏布局了（图 7-11）。没有必要调整 about 和 credits 这两个 div 的宽度或者边距，因为 about 栏已经被设置成为自适应，它会自动缩放为新增的第三栏留下足够的空间。不论父容器是固定宽度，还是自适应可视区域，这些设置都能够正常工作。



图 7-11 about 容器会自动调整宽度，确保父容器可以容纳新加入的栏目

这种易于装载和拆卸栏目而不影响周围元素的各项设定的能力，能够让 CSS 更加简洁，在实际项目中，这也能让工作更加轻松。举例来说，如果项目需要新增一个边栏，但是只显示在特定的页面比如主页、新闻页或者关于我们页中。那么，创建一个包含新闻边栏的三栏布局就跟两栏布局一样简单了，也能够大大节省代码量。因为页面中的元素会自动调整确保新闻边栏能够顺利加入，这样，为两栏布局和三栏布局创建不同的 CSS 就没有任何必要了，再也无需书写如下的代码：

```
body.two-col #content { width: 75%; }
body.three-col #content { width: 60%; }
body.two-col #nav { width: 25%; }
body.three-col #nav { width: 20%; }
```

7.2.3 重定栏目顺序

另外一个使用弹性布局模型的好处是可以在不修改 HTML 代码的情况下很容易地重新排列栏目的顺序。这里有几个属性是与这个特性相关的。

最简单的属性是 box-direction。举例来说，如果希望边栏在左边而不是右边，只需要在 #content 规则中设置 box-direction 属性为 reverse，然后将边栏的左边距设置成为右边距：

```
#content {
    display: -moz-box;
    display: -o-box;
```

```
    display: -webkit-box;
    display: box;
    -moz-box-orient: horizontal;
    -o-box-orient: horizontal;
    -webkit-box-orient: horizontal;
    box-orient: horizontal;
    -moz-box-direction: reverse;
    -o-box-direction: reverse;
    -webkit-box-direction: reverse;
    box-direction: reverse;
}
#content-secondary {
    width: 16em;
    margin: 0 40px 40px 0;
}
```

这使得浏览器还是会水平排列这些子元素，但是会按照自右至左而不是自左至右的顺序。content-main 这个 div 在 HTML 代码里面先出现，所以浏览器首先渲染它，但是会先从右边开始，然后再把 content-secondary 这个 div 放在它的左边（图 7-12）。如果这些子元素是垂直而不是水平排列的话，设置 box-direction:reverse 的结果就会让浏览器按照自下而上而不是自上而下的顺序排列元素。

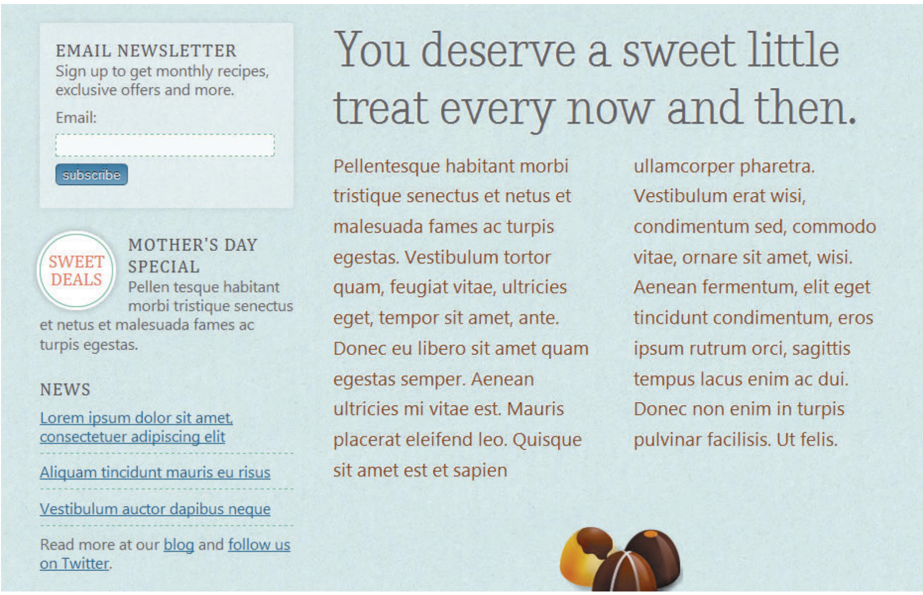


图 7-12　设置了 box-direction 为 reverse 以后，浏览器按照自右至左的顺序排列元素

如果想更细致地控制每个子元素的排列方式，可以使用 box-ordinal-group 属性。当 box-ordinal-group 属性设置为 1 的时候，元素将会在第一位出现，设置为 2 的时候，将会在第二位，以此类推。这意味着在本书的例子中，如果想把 learn-more 元素放在最左边而不

是中间显示的话，只需要为底部的三个子元素设置如下的 box-ordinal-group 值：

```
#about {
   -moz-box-flex: 1;
   -o-box-flex: 1;
   -webkit-box-flex: 1;
   box-flex: 1;
   -moz-box-ordinal-group: 2;
   -o-box-ordinal-group: 2;
   -webkit-box-ordinal-group: 2;
   box-ordinal-group: 2;
}
#credits {
   -moz-box-ordinal-group: 2;
   -o-box-ordinal-group: 2;
   -webkit-box-ordinal-group: 2;
   box-ordinal-group: 2;
   width: 10em;
   margin: 0 0 40px 40px;
}
#learn-more {
   -moz-box-ordinal-group: 1;
   -o-box-ordinal-group: 1;
   -webkit-box-ordinal-group: 1;
   box-ordinal-group: 1;
   width: 10em;
   margin: 0 0 40px 40px;
}
```

当然，要把 learn-more 这个 div 的左边距改成右边距：

```
#learn-more {
   -moz-box-ordinal-group: 1;
   -o-box-ordinal-group: 1;
   -webkit-box-ordinal-group: 1;
   box-ordinal-group: 1;
   width: 10em;
   margin: 0 40px 40px 0;
}
```

Reverse 存在的问题

请慎用 box-direction:reverse 设置，这会产生一些意料之外的问题。举例来说，具体的内容也会自右至左而不是自左至右。另外一些属性，比如控制摆放位置或者对齐方式的 box-align 或者 box-pack（本章稍后会提到这两个属性），会因为 reverse 而产生不正常的表现。如果你发现某个元素的表现和期望正好相反的时候，检查一下是否使用了 reverse 设置，那可能是问题产生的原因，好好测试吧！

说明　至此为止的所有修改都在 flex-box_2.html 页面里，它在你下载的本章的示例文件中。

现在 learn-more 这个 div 已经在并列的几个元素中有了较低的 box-ordinal-group 值，浏览器会优先渲染它。又由于元素的排列被指定为水平自左至右排列（box-orient 被设置为 horizontal，box-direction 默认的设置为 normal），这就会让 learn-more 元素渲染在左边的第一位（图 7-13）。浏览器接下来会把 about 和 credits 这两个 div 依次放在右边。由于这两个 div 的 box-ordinal-group 值是相等的，浏览器会通过它们在 HTML 源代码中的顺序来决定它们出现的先后：代码在前的 div 优先排列，这和那些没有指定这个值的 div 的排列次序是一样的。



图 7-13　通过设置 box-ordinal-group 的值，即使元素的 HTML 代码位置在其他元素中间，也能优先显示

能够无视元素的源代码顺序而任意控制元素的排列位置，这个功能确实非常强大。能够让最重要的内容始终居于 HTML 代码的最前部分，但是又可以做到这些内容不会显示在页面的最前面。这对于页面的残障人士可访问性，多设备的兼容性以及搜索引擎优化等都能起到很大的作用。

7.2.4　多栏等高

弹性布局模型的另外一个有点微不足道的好处就是能够很轻易地创建高度相等的多个栏目。

在 CSS 问世前，很多设计师（包括我）认为创建高度相等的栏目的最佳方法是使用 table。在 table 的同一行中的单元格的高度总会自动匹配的，但是依次排列的独立 div 没有理由这样做。为了让它们能够等高显示，设计师和开发者用了很多 CSS 小技巧来解决这个问题。这些技巧可能包括添加一个或者多个包裹容器，或者使用背景图片（甚至有时候只需要一个简单的背景色，或者仅仅只需要边框而不需要任何背景色），以及相对复杂的 CSS。

弹性布局模型如果要创建多栏等高的效果的话，还是需要添加一个额外的包裹容器，但是不需要任何图片（这样减少 HTTP 请求后页面加载会更快），它的 CSS 规则也非常简单，只需要将包裹这些栏目的父容器的 box-align 属性设置成 stretch。stretch 是 box-align 的默认值，所以如果为本章的例子中的两个主要内容区域设置背景色的话，你立刻就可以看到效果。

```
#content-main {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
    margin-bottom: 40px;
    background: #dcdcdc;
}
```

```
#content-secondary {
    width: 16em;
    margin: 0 40px 40px 0;
    background: #fcc;
}
```

保存页面并在 Firefox、Safari 或 Chrome 中预览。边栏的粉色背景一直延伸到了页面的底部，而页面的高度是由内容较多的另外一个灰色背景的栏目决定的（图 7-14）。



图7-14　两个内容区块延伸到相同的高度，这是因为 `box-align` 的值默认就是 `stretch` 的缘故（另见彩插图7-14）

再次强调，之所以会这样，是因为 box-align 的值默认为 stretch，如果为#content 强制加上这条规则，那效果也是一样的：

```
#content {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    -moz-box-orient: horizontal;
    -o-box-orient: horizontal;
    -webkit-box-orient: horizontal;
    box-orient: horizontal;
    -moz-box-direction: reverse;
    -o-box-direction: reverse;
    -webkit-box-direction: reverse;
    box-direction: reverse;
    -moz-box-align: stretch;
    -o-box-align: stretch;
    -webkit-box-align: stretch;
    box-align: stretch;
}
```

box-align 属性控制了子元素们相互排列时垂直于排列方向的对齐方式。如果像例子中那样，子元素是水平排列的，box-align 控制的就是垂直方向的对齐方式，如果子元素是垂直排列的，那 box-align 控制的就是水平方向的对齐方式。

表 7-1 总结了 box-align 的可能值。请注意，在这里 start 和 end 的定义，当元素设置了 reverse 方向时，是相反的。举例来说，水平排列的元素，如果设置了 reverse，那么它的 box-align 如果设置成 start 时，它将会从底部开始对齐，而不是通常那样从顶部开始对齐。然而，现有浏览器当前并没有遵循这一规范。

表7-1　box-align的取值

取　　值	水平子元素	垂直子元素
start	顶部对齐	向左对齐
end	底部对齐	向右对齐
center	垂直居中对齐（其顶部和底部的空间相等）	水平居中对齐（左边和右边的空间相等）
baseline	每个区块的首行文本基于文本的基本线对齐，然后以最高的子元素为盒子的顶部边界	与center相同*
stretch	垂直拉伸以符合盒子的高度	水平拉伸以符合盒子的宽度

* Firefox对垂直区块的baseline取值等同于start，而不是center。

7.2.5　水平和垂直居中

box-align 属性不仅能够实现等高的多个栏目，而且也能让垂直居中——这是 CSS 2.1 几乎不可能完成的任务——变得容易实现，即使在父元素和子元素的高度都未知的情况下。另外的一个属性 box-pack 也让水平居中变得不再需要传统的复杂技巧。

让我们在页面的头部区域尝试这两种居中特性。对于从第 6 章就开始使用的这个页面来说，此处做了一点轻微的改动：header 容器现在包裹了 logo 图片和搜索表单，而 nav-main 这个容器则分离出去了。

1. 让 logo 和搜索表单垂直居中

包裹容器里面仅包含 logo 和表单的时候，两个元素的相对垂直居中更加容易实现了。

首先把头部区域的 `display` 属性设置成 `box` 使其进入弹性布局：

```
#header {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    padding: 20px 0;
}
```

如先前所说，即使不设置 `box-orient` 属性的值，弹性容器的子元素也会默认水平排列。因此上面的这段 CSS 代码会使得 logo 和搜索表单排列在同一行（图 7-15）。



图 7-15　logo 和搜索表单成为弹性容器后，会按照水平顺序排列

为了让搜索表单显示在屏幕的右方而不是顶着 logo 显示，需要让它延伸以填满头部除了 logo 宽度以外的所有空间。同时再设置 `text-align` 的属性值为 `right`，这样就保证了搜索表单里面的内容都是右对齐的。请添加如下规则来完成上述工作：

```
#form-search {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
    text-align: right;
}
```

现在在 Safari 和 Chrome 里面搜索表单已经如我们所希望的那样处于屏幕的右侧了（图 7-16）。但是在 Firefox 上还有点问题。这是因为 Firefox 使用了头部区域的固有大小，从而使得头部区域的宽度只有它内部元素的宽度之和。对于子元素来说，这个判断是合适的，但是对于父容器，Firefox 这么做是有问题的。这个问题很容易修改，只需要将头部区域的宽度设置成百分之百：

```
#header {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    width: 100%;
    padding: 20px 0;
}
```

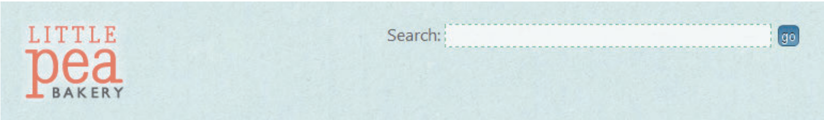


图 7-16　在基于 Webkit 的浏览器中，搜索表单已按照期望填满了剩余区域，并且显示在右上侧

这会让 Firefox 停止缩小头部区域的宽度，而会让头部区域延伸填满整个包裹容器。现在，Firefox 的搜索表单也能正确地显示在右上侧了。

logo 和搜索表单在水平方向上已经处于预期的位置了，现在来看看如何让它们在垂直方向上也能如预期排列。唯一需要做的就是将头部区域的 box-align 属性设置成为 center。

```
#header {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    -moz-box-align: center;
    -o-box-align: center;
    -webkit-box-align: center;
    box-align: center;
    width: 100%;
    padding: 20px 0;
}
```

通过这条规则，logo 和搜索表单就能够在头部区域中垂直居中对齐（图 7-17）。不管表单的字体大小如何增长，或者产生换行，浏览器始终会自动调整保证元素的垂直居中对齐。

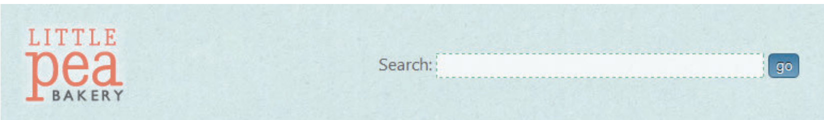


图 7-17　将头部区域的 box-align 设置成 center 可以让它的子元素垂直居中对齐

2. 让导航条水平居中

现在，将注意力转移到页面中的另外一个元素，导航条。为了让它在包裹容器中水平居中，设置 box-align:center 是不管用的——因为它只对水平排列的元素在垂直方向的对齐才能起作用，或者是垂直排列的元素在水平方向的对齐。我们需要一个新的属性达到这个效果。

消失的元素

如果不是在整个页面布局而仅仅是在页面的某几个元素上使用弹性布局，那很有可能某个元素被同时赋予 display:box 属性和 float 或者 overflow 属性。如果是这样的话，可能会带来很大的问题。

> Safari 和 Chrome 存在一个缺陷，就是会让设置了 float 属性的弹性布局元素从页面上消失。虽然在弹性布局元素上是不可以使用 float 的，但让元素消失也不应该——而是应该忽略这条不合理的属性。对于 Firefox，如果弹性布局元素被赋予 overflow 属性时，也会导致这种消失。
>
> 这些缺陷使得混合弹性布局和传统布局方式（为了兼容旧式浏览器）变得越发困难。举例来说，例子页面中的导航条，每个 li 元素都是设置成浮动的，这使得导航链接在所有浏览器中水平排列。如果期望 ul 元素能够包含它的浮动子元素 li 的话，就必须使用常见的浮动容纳技术，或者让 ul 也浮动，或者为 ul 设置 overflow: auto 或 overflow: hidden。但是由于 ul 元素现在是弹性布局的子元素，对它设置浮动会让它在基于 Webkit 的浏览器中消失，但是设置 overflow 又会让它在 Firefox 中消失。

正好 box-pack 属性能够满足以上的要求。通过使用它，可以让 ul 元素移动到 nav-main 这个 div 的中间。为此，首先要将 nav-main 这个 div 设置成为浮动布局然后将它的 box-pack 属性设置成 center。CSS 规则如下所示：

```
#nav-main {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    -moz-box-pack: center;
    -o-box-pack: center;
    -webkit-box-pack: center;
    box-pack: center;
    overflow: auto;
    margin: 0 0 20px 0;
}
```

接下来，还得为 Firefox 做一系列的工作。给 div 增加属性 width:100%来保证它会填充整个容器。同时移除 overflow:auto 声明来修复会导致 div 消失的 Firefox 缺陷（参见之前的“消失的元素”）。

```
#nav-main {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    -moz-box-pack: center;
    -o-box-pack: center;
    -webkit-box-pack: center;
    box-pack: center;
    width: 100%;
    margin: 0 0 20px 0;
}
```

现在保存页面并在 Firefox、Safari 和 Chrome 中观察效果。导航条已经在页面中水平居中显示了（图 7-18）。将 box-pack 设置成为 center 会让浏览器利用所有的水平空间并让元素居中

显示。

说明　完成版页面所展示的效果代码都在 flex-box_final.html 页面里，它在你下载的本章的示例文件中。

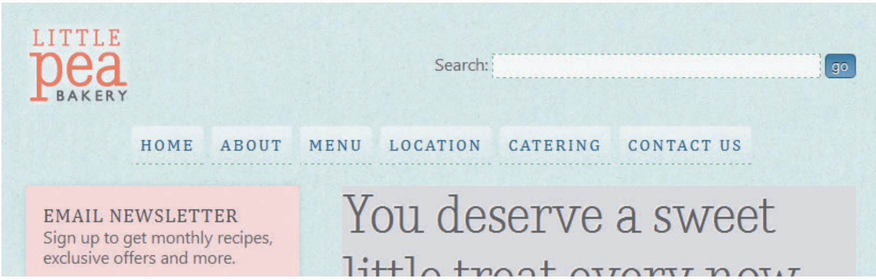


图 7-18　通过设置 box-pack 为 center 可以让导航条居中显示

换行显示子元素

如果有足够空间的话，弹性布局的子元素默认是不会换行显示的——但是它们会覆盖内容。这跟浮动的工作机制正好相反。虽然浮动布局将超出宽度的子元素自动换行显示的特性有时令人无语，总的来说它还是件好事。举例来说，导航条中的 li 元素是浮动定位的，所以当窗口太窄而无法全部显示它们的时候，它们就会换行显示，这样至少保证了所有的导航链接任何时候都在可视区域。如果 ul 元素是弹性布局的，然后子元素是水平排列的，这时候 li 元素是不会换行的，它们会顽固的保留在一行，即使是窗口窄到无法完整显示它们。

在这种情况下，如果需要换行的话，可以设置 box-lines 属性的值为 multiple，不幸的是，目前还没有任何浏览器可以支持。因此，目前不要对任何需要换行的元素使用弹性布局。

表 7-2 总结了 box-pack 属性所有可能的值。如同 box-align 一样，start 和 end 的定义在容器被设置了 reverse 之后将会是相反的。至少 Firefox 是遵循这个原则的，虽然基于 Webkit 的浏览器没有这样做。

表7-2　box-pack的取值

取　值	水平子元素	垂直子元素
start	位于盒子左侧	位于盒子顶部
end	位于盒子右侧	位于盒子底部
center	水平居中对齐（左边和右边的空间相等）	垂直居中对齐（其顶部和底部的空间相等）
justify*	每个子元素之间的空间相等，首元素左边和末元素右边无剩余空间。如果只有一个元素，那么与start相同	相同

* Firefox不支持box-pack属性的justtify值，它会直接把它当做start来对待。

7.3　实际情况：现实可用部分

不幸的是，本章上述的大部分工作目前来看还仅仅是一种蓝图——实际上弹性布局模型还没法运用到实际工作中。因为 IE、Opera 还有其他一些浏览器都不支持它，所以眼前这个布局看起来真是乱七八糟（图 7-19）。这可不是使用了 `border-radius` 之后在 IE 上无法显示圆角之类的问题，这个 CSS3 属性会影响到整个页面的布局，它带来的问题可比简单的视觉效果大多了。

最新版本的 Modernizr 能够检测浏览器是否支持弹性布局模型，这样就可以用它来为不支持的浏览器提供另外一种布局方式如浮动等。显而易见的是，这与使用弹性布局模型的初衷是相背离的，如果已经有了适应任何浏览器的浮动布局模型，那为什么还要花时间去创建弹性布局模型呢？



图 7-19　在不支持的浏览器（如 IE 8）上，页面惨不忍睹

别泄气！现实情况中弹性布局模型还是有用武之地的。在上一节讨论中，我们说过它可以用来创建全页面的多列布局因为它能让布局元素很好地弹性自适应，所以这将是我们未来构建布局的方式。在实际的情况中也有可能运用这种布局。

举例来说，弹性布局模型很容易将导航栏水平居中，而这个特性对于不支持的浏览器的降级处理很好。因为此处并没有使用弹性布局模型让每个单独的链接水平排列，而仅仅是让 `ul` 元素水平居中，所以 IE 和 Opera 还是能看到一个水平排列的导航列表，只是没有水平居中而已，在大多数情况中，这是一个完全可以接受的降级体验，同时也让你无需用复杂的方式实现一个浮动定位的导航栏的水平居中。

除了水平居中的例子外，下面还有一些其他的实际运用弹性布局模型的例子。

7.3.1　弹性表单布局

使用弹性布局模型的一个重要方式是用它来布局表单元素。请在浏览器中打开练习文件 form_start.html。这跟第 6 章所使用的文件是相同的，采用了浮动布局，这保证了在所有浏览器上的可用性。

当实际媒体的可视区域小于等于 500 像素时，label、文本区域和邮件订阅按钮排列在一行（图 7-20）。但是表单并没有拉伸填充整个父容器的区域。如果文本区域能够调整拉伸程度以自动填满剩余空间的话，视觉效果会更好。这在不使用弹性布局模型的情况下，其他 CSS 技术也能实现这一点，但是需要一定数量的嵌套 `div` 以及比较复杂的定位规则（参见 http://friedcellcollective.net/outbreak/2009/10/04/fluid-searchbox）。

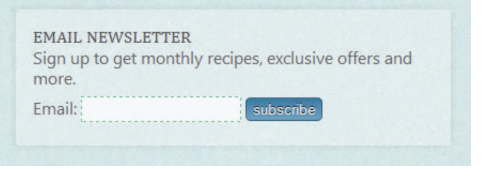


图 7-20　文本区域是定宽的，看起来没有问题，只是没有拉伸填满整个空间

使用弹性布局模型将会使文本区域的拉伸非常容易实现。首先找到页面代码 `head` 区域内的样式的最后 `max-width: 550px` 定义，然后删除下面两条：

```
#form-newsletter * { display: inline; }
#form-newsletter input[type=text] { width: auto; }
```

这样就能让元素的定义回到默认状态，接下来将展示新的弹性布局模型是如何轻易地实现所需要的样式效果的。

接着将表单转成弹性布局：这将使它的子元素默认水平排列。同时也必须将它的宽度设置成 100%，这样才能保证它会拉伸填满父容器。请添加如下的一条新的规则：

```
#form-newsletter {
    display: -moz-box;
    display: -o-box;
```

```
    display: -webkit-box;
    display: box;
    width: 100%;
}
```

接下来，为表单元素添加一定量的边距：

```
#form-newsletter * {
    margin-right: 3px;
}
#form-newsletter :last-child {
    margin-right: 0;
}
```

现在，整个表单看起来比之前好多了，如图 7-20 所示，但文本区域还没有拉伸。这是因为表单元素目前还保持它们的固有大小。对于 `label` 和按钮来说，这是很完美的，本来就期望它们的宽度和内部文字内容保持适应，但是对于文本区域而言不是这样。请添加以下的 CSS 规则：

```
#form-newsletter input[type=text] {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
}
```

保存页面并在 Firefox、Safari 和 Chrome 上观看。将窗口的宽度缩小到 550 像素以激活媒体查询，然后再看邮件订阅表单的表现，继续缩小窗口宽度，就能够观察到文本区域始终能够填充可用空间（见图 7-21）。即使父容器被设置成为固定宽度，或者用户使用了特定的字体大小从而改变了 label 和按钮的大小，文本区域始终能够通过自动调整使用剩余的可用空间。

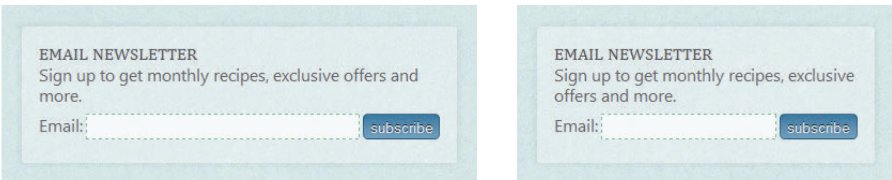


图 7-21　不论父容器的大小是多少，文本区域始终能够自动拉伸

页面头部的搜索字段也能完美地使用这个效果，可以组合 `label` 文本、文本字段、提交按钮、甚至一个用于高级搜索的链接，然后让它们全部排于一行，并使用文本字段自动拉伸填满可用空间。表单元素有些时候在对齐上很难处理，此时 `box-align` 和 `box-pack` 会让工作变得简单。

降级处理

在 Firefox 和基于 Webkit 核心的浏览器上订阅表单看起来非常完美，但是在 Opera 和 IE 9 上却有点糟糕，这些浏览器不支持弹性布局。此时表单中的每个元素都会独占一行（图 7-22）。



图 7-22　在不支持的浏览器中，每个元素会独占一行

尽管没法为这些不支持的浏览器提供特有的样式而不影响 Firefox 和 Webkit，但在这个例子中还是能够为所有的浏览器提供一个降级处理的样式，然后再在 Firefox 和 Webkit 中重载样式而让不支持的浏览器无法感知。

在不支持弹性布局的浏览器中如果需要让表单元素排在一行的话，需要为#form-newsletter input[type=text]规则添加 width:auto 属性，然后为#form-newsletter * 规则添加 display: inline 属性：

```
#form-newsletter * {
    display: inline;
    margin-right: 3px;
}
#form-newsletter input[type=text] {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
    width: auto;
}
```

这实际上就是把之前去掉的两条规则给恢复了，这样在 Opera 和 IE 9 上表单元素重新又排在一行了。这里本来可以直接保持原状的，之所以先去掉再加上是为了让这部分工作产生的效果更加明晰。

这样做唯一的问题就是将表单元素设置成 display:inline 之后，会重载之前对弹性布局的定义，从而导致弹性布局失效。为了避免这一点，需要在 display:inline 之后再添加对 display:box 的定义，包括带浏览器前缀的定义：

```
#form-newsletter * {
    display: inline;
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    margin-right: 3px;
}
```

设置 display 的 box 值以后，inline 值就被重载掉了，这样弹性布局的定义就不会失效。不支持弹性布局的浏览器自然也无法理解 box 这个值，这样它就会继续使用 inline 这个值，之后一系列的 display 定义也由于同样的原因被忽略了。

通过这些样式，使用不支持弹性布局的浏览器的用户也能看到如图 7-20 所示的同样内容的表单布局，就是在第 6 章中所使用的适合所有浏览器的样式，而在支持弹性布局的浏览器上将会看到增强版本的表单样式，如图 7-21 所示。尽管缺少浏览器支持的问题可能会导致实际项目中无法将弹性布局应用于整个页面布局，但是在之前的这些例子里面，没有理由不应用这个布局技术做一些小的渐进增强。

说明　完成版页面所展示的效果代码都在 form_final.html 页面里，它在你下载的本章的示例文件中。

7.3.2　底部区域黏附

使用弹性布局模型，同时又不破坏不支持的浏览器的另外一个好的例子是用它来创建底部区域黏附效果。所谓底部区域黏附通常是指页面的底部区域固定在整个可视区域的底部，即使页面的主体内容的长度不足以将底部区域推到可视区域的底部（图 7-23）。不使用 CSS3 的话，也能实现这个特性，但是当然，这会比较复杂。（参见 www.cssstickyfooter.com/using-sticky-footer-code.html，这里描述了一种方法，也有其他方法的引用链接。）

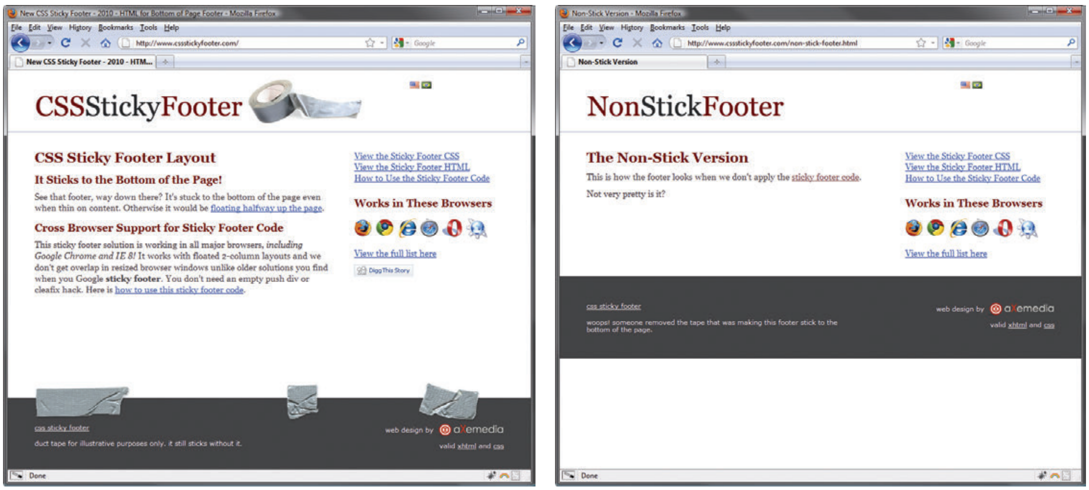


图 7-23　左边的页面中，底部区域始终显示在可视区域的底部，右边的页面中底部区域只是显示在内容区域的底部

使用弹性布局模型来创建底部区域黏附效果的关键是使用 `box-flex` 属性让底部区域之前的 `div` 弹性自适应。这将会让这个 `div` 自动拉伸填充可视区域中的所有可用空间。如果此时没有足够的拉伸空间，换句话说，页面的长度其实已经够长，甚至超过了可视区域，那么在底部区域之前的 `div` 会保持自己的固有高度，底部区域也会照常直接显示在其后。

如果想自己试试的话，只需要在编辑器中打开文件 sticky-footer.html。这跟第 6 章的页面是一样的，只是增加了一个名为 content 的 `div` 来包裹 content-main 和 content-secondary。同时也移除了一些内容来使页面变得比较短，此时底部区域紧贴在内容区域的下方，和可视区域的底部之间还有一定距离（图 7-24）。

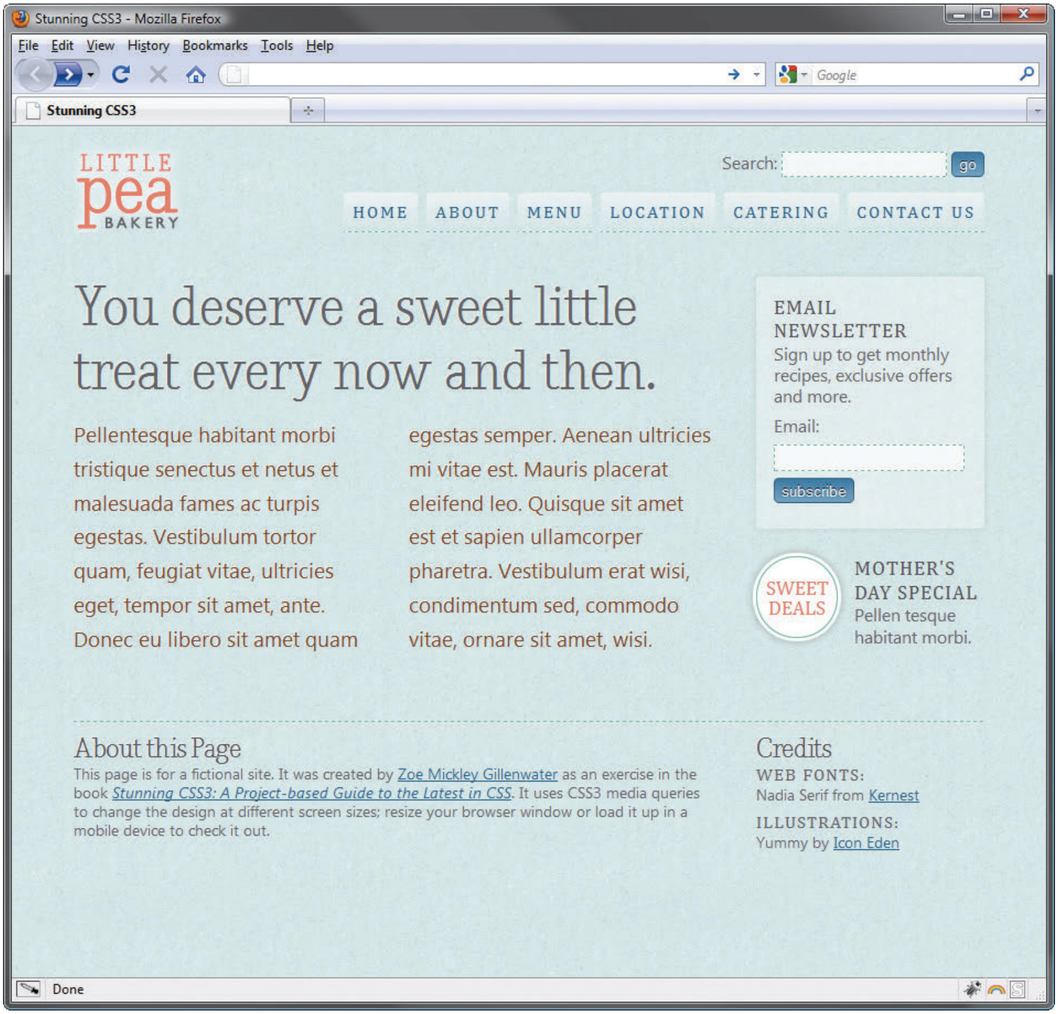


图 7-24　底部区域正好显示在内容区域的下方，而不是可视区域的底部

如果希望不管在什么情况下整个页面的布局至少要和可视区域一样高，那么首先要保证 `body` 和 `html` 这两个元素能够拉伸到和可视区域一样高，即使在它们不包含任何内容的情况下。

为 `html` 元素创建规则，将 `height` 属性设置为 `100%`，然后为已经存在的 `body` 规则也添加 `height:100%`属性：

```
html {
    height: 100%;
}
body {
    height: 100%;
    margin: 0;
    padding: 0;
    background: url(images/background.jpg);
    color: #666;
    font-family: "Segoe UI", Segoe, Calibri, Arial,
    ¬ sans-serif;
    font-size: 100%;
    line-height: 1.6;
}
```

接下来，使用 display 属性将包裹 div 设定成弹性布局，然后使用 box-orient 属性设定它的子元素垂直排列。然后将其最小高度设置成 100%，这样它会始终保持至少和可视区域一样高：

```
#wrapper {
    display: -moz-box;
    display: -o-box;
    display: -webkit-box;
    display: box;
    -moz-box-orient: vertical;
    -o-box-orient: vertical;
    -webkit-box-orient: vertical;
    box-orient: vertical;
    min-height: 100%;
    margin: 0 6%;
}
```

这将使得包裹容器的三个子元素——header、content 和 footer——垂直排列。这本来就是它们的默认排列方式，所以这些规则不会带来任何浏览器下的外观改变，不管它们是否支持弹性布局。但是添加这些规则的目的是要让包裹容器的任意子元素具有弹性自适应的能力。

而在这个例子里面，被期望具有弹性自适应能力的子元素就是 content 这个 div，因此可以为它添加新的规则，将 box-flex 属性设置成 1：

```
#content {
    -moz-box-flex: 1;
    -o-box-flex: 1;
    -webkit-box-flex: 1;
    box-flex: 1;
}
```

现在，扣除了头部区域和底部区域的固有高度之后，不管包裹容器里面还剩余多少内容区域的可用高度，这三个区域的高度之和总会至少等于可视区域的高度。因此，底部区域将会定位在可视区域的底部，而不紧贴内容区域（图 7-25）。

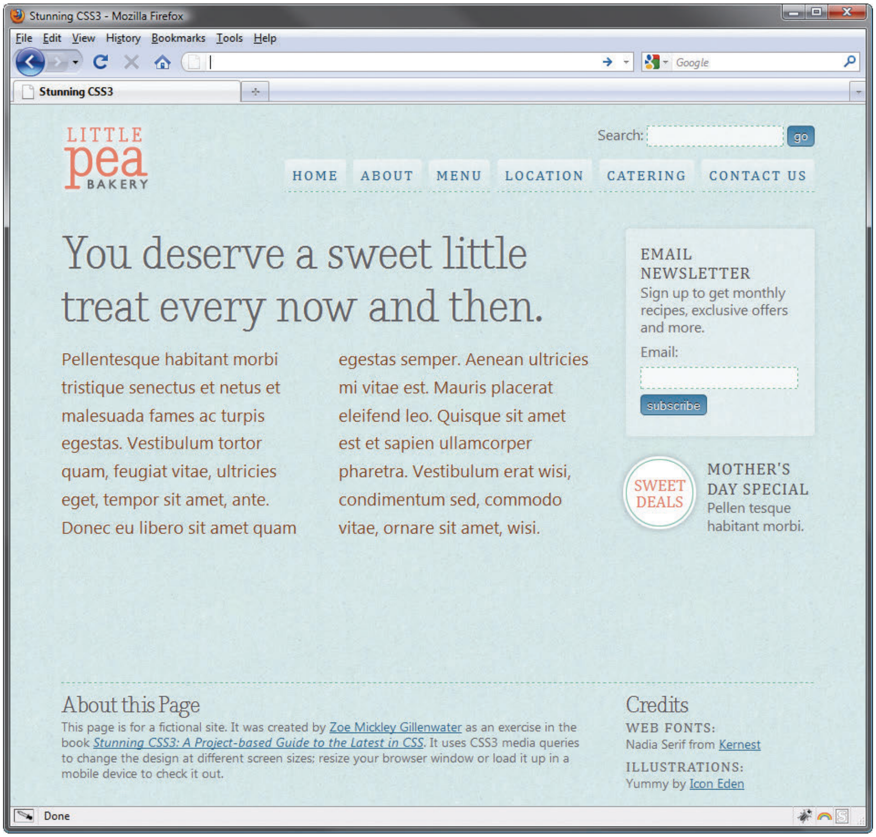


图 7-25 底部区域正好显示在可视区域的底部，而不是内容区域的底部

不支持弹性布局的浏览器会简单地忽略这些规则，所以页面还是如前所示，效果没有改变（就如当前大部分页面所表现的那样）：底部区域正好紧贴在内容区域的下方。使用这些浏览器的用户完全不知道他们所看见的内容显示方式和其他人会有所不同。底部区域黏附的效果，是对使用高级浏览器的用户的额外装饰性加分。

说明 完成版页面所展示的效果代码都在 footer-queries_final.html 页面里，它在你下载的本章的示例文件中。

弹性布局模型的来龙去脉

你可以在 www.w3.org/tr/css3-flexbox 找到弹性布局模型（Flexible Box Layout）。比起 CSS2.1 所提供的布局技术，弹性布局模型所创建的布局在处理对齐和间距等问题上具有更强大的能力。

如果要使用这个布局模型，只需要将一个包裹容器的display属性设置成box，这是CSS3引入的一个新值。接下来就可以使用一系列的新属性来控制包裹容器内的子元素的排列布局方式（参加表7-4）。

除了本章中已经给出的例子以外，弹性布局模型还可以用于以下场景。

- 使用box-direction或者box-ordinal-group属性让重要的内容始终居于HTML代码的前面，但是无需让它在视觉上显示在页面的前面。
- 使用box-ordinal-group让博客的某个帖子置顶，即使这个帖子在HTML代码中并不是第一条。
- 创建一个全宽的水平导航，使用box-flex让每个按钮都弹性自适应宽度。
- 可以让导航条中处于鼠标悬停状态或者当前选中状态的按钮自适应占据所有的剩余可用空间，这个效果可以参见www.IE 7nomore.com/fun/flexiblenav。
- 图片库效果，让每一行不同高度的缩略图垂直居中对齐。
- 视频播放器控制按钮效果，进度条是弹性自适应的，占据了扣除播放控制按钮之外的所有剩余空间；参见http://clubajax.org/css3-layouts-the-flexible-box-model-basics。
- 让内容图片和对应的图片描述文字相邻垂直居中。

这些例子的大部分在实际项目中还不能使用，因为还存在很多的浏览器不兼容的情况以及其他缺陷，但是如果为特定的浏览器创建一个应用的话，比如iOS上的Safari，还是值得一试的。

表7-3　弹性布局模型的浏览器支持度

IE	Firefox	Opera	Safari	Chrome
不支持	部分支持，需要-moz-前缀	不支持	部分支持，需要-webkit-前缀	部分支持，需要-webkit-前缀

表7-4　弹性布局模型的属性

属　　性	作　　用
box-orient	使区块在垂直或水平方向上堆积
box-direction	改变默认的由上至下或由左至右的堆积顺序
box-ordinal-group	按顺序指定需要显示的区块编号，比box-direction提供更强的堆积顺序控制力
box-flex	使区块弹性地填充其他区块定位后的可用剩余空间
box-flex-group	使用编号将弹性区块进行分组，当有剩余空间时浏览器能够首先调整位于第一分组内的区块宽度，如还有额外空间再考虑第二分组，以此类推
box-align	用与区块相对的排列方式控制其对齐与定位，包括将其拉伸来填充可用空间以创建出等宽或等高的区块
box-pack	用与区块相同的排列方式来控制其对齐与定位
box-lines	如果子元素无法保持在同一行内，可使其折行，类似设置浮动之后的折行，而非溢出

7.4　弹性布局模型的替代品

弹性布局模型并不是 CSS3 中唯一的布局工具。还有其他的布局系统也正处于工作草案，鲜有浏览器支持。这里会先讨论一个支持度最好，用途广泛的 `box-sizing` 属性，然后再看看有什么其他可以期待的。

7.4.1　box-sizing属性

在 W3C 的传统 CSS2.1 盒模型中，一般是通过声明 `width` 和 `height` 的值来控制内容区域的宽度和高度，然后再附加上内边距和边框等。这个在术语上被称为内容盒模型。通常设计师和开发者会对此习以为常，不加思索。其实这个模型在某些情况下并不是十分方便，比如为一个容器的宽度和内边距设置不同衡量单位时，例如宽度是百分比但内边距是像素值。

举例来说，当希望为页面中的三个产品宣传容器设置不同的背景色，但又设置一定像素的内边距来保证容器内的内容不会完全遮盖住背景色时，该怎么做？当希望为每一个容器添加边框的时候该怎么做？

为了实践本节所将要提到的内容，请打开本章练习文件中的 box-sizing_start.html。这跟第 6 章中的页面是一致的，只是移除了媒体查询的部分让练习更简单一些。

找到 116 行已经存在的`.feature` 规则，在页面 `head` 区的样式表代码的中间位置。修改 `padding` 值为每个容器增加底部和周边的内边距：

```
.feature {
    float: left;
    width: 30%;
    margin: 0 4.5% 0 0;
    padding: 130px 15px 5px 15px;
    background-repeat: no-repeat;
    background-position: top center;
}
```

接下来，增加背景色和边框：

```
.feature {
    float: left;
    width: 30%;
    margin: 0 4.5% 0 0;
    padding: 130px 15px 5px 15px;
    border: 1px dashed #3C9;
    background-color: hsla(0,0%,100%,.3);
    background-repeat: no-repeat;
    background-position: top center;
}
```

保存页面并在浏览器中预览。第三个容器将会换行显示（图 7-26）。这是因为，每个容器本身要占据 30%的空间，然后要加上 2 像素的边框，以及 30 像素的内边距，此时如果再为第一个和第二个容器设置 4.5%的外边距的话，三个容器所占据的空间的总和将会超过 100%。比 100%

大多少呢？这目前还是未知的，在不同显示设备上会有所不同，重要的是混合设置像素和百分比将会使找到一个合适的宽度变得不太可能。

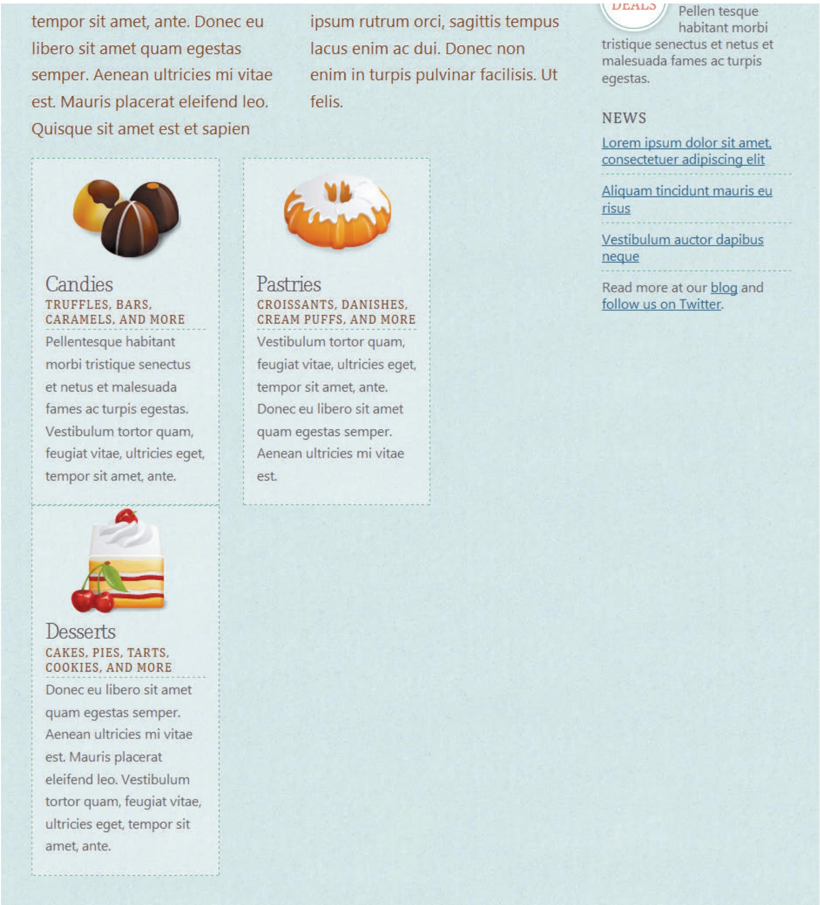


图 7-26　第三个容器被换行显示了，因为总的宽度已经超出了 100%

为了修改这个问题，就必须减小每个容器的宽度。当把宽度从 30%变成 25%时，在特定的窗口大小上，这三个元素又排成一行了——但是在比较窄的窗口上，第三个容器还是会换行显示（图 7-27）。另外，在非常宽的窗口宽度上，在第三个容器的右边将会有一段空白的空间。它们不能够总是很好地适应宽度。

除了改变容器的宽度之外，另外还有一个方法是在三个容器的内部各嵌套一个 `div`，然后将边框和内边距都增加到这个嵌入的 `div` 上，这样，外部的三个容器还是精确地各占 30%的宽度，边框和内边距的宽度都属于内部嵌入的 `div`。嵌入一个这样的 `div` 不会有多麻烦，但是在一个比较复杂的设计中，嵌入的 `div` 的数量可能会稍微有点多，这将导致 HTML 以及 CSS 的开发时间和文件大小都会大大增加。

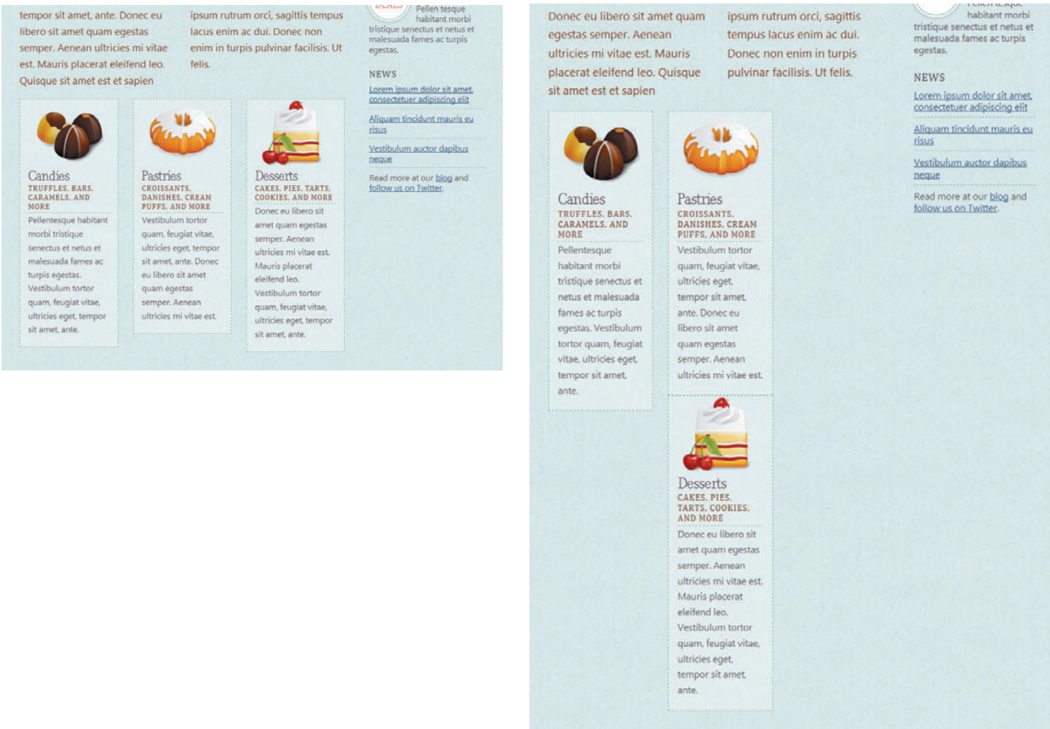


图 7-27　减少宽度的方法对于大一点的可视区域还有作用，但是对于很窄的窗口就不行了

CSS3 引入了一个更有效率的方法，无需修改 HTML 代码，这就是新的 `box-sizing` 属性。这个属性采用了一种新的边框盒模型而不是传统的内容盒模型。当一个容器使用边框盒模型时，浏览器会从容器的内部裁剪空间给内边距和边框使用，而不是把它们的空间附加上去（图 7-28）。这样容器所实际占用的宽度和声明的宽度就一直保持一致了。

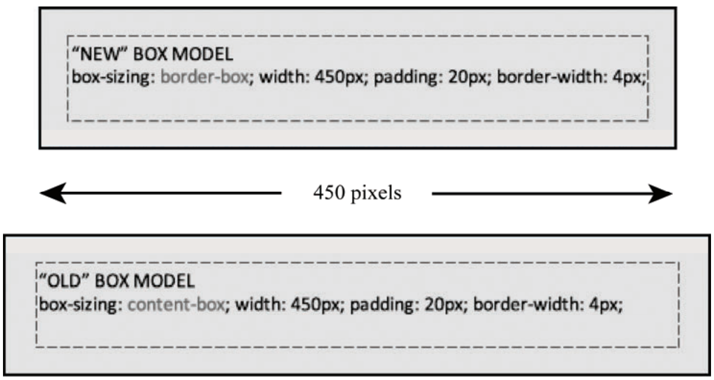


图 7-28　内容盒模型和边框盒模型的区别

更复杂的宽度计算

在这个混合单位例子中，另外一个将要实现的有用的特性是 calc 功能。它能够使用在所有接受长度值的地方，比如宽度、外边距，它接受一个计算之后的结果。举例来说，在上述的示例页面里，可以设置每个容器的宽度为：calc (30%-32px)。

Calc 功能最酷和最强大之处在于能够使用比上述值更复杂的计算等式来获得长度值，这比 box-sizing 所能够处理的情况要更多一些。不好的一面在于计算等式里面的数值只能够硬编码，如果以后要改变某个数值比如内边距，就不得不在 calc 里面做联动改变。

不幸的是，本书撰写之时，还没有浏览器可以支持这个功能。即将到来的 Firefox 4 也许会，同时也支持 min 和 max 功能，但是这些功能在 Firefox 目前发布的 beta 版本中尚未得到支持，如果要获取更多关于 Firefox 对于 calc 功能实现以及相应的一些例子，请参见 http://hack.mozilla.org /2010/06/css3-calc。W3C 官方的定义请参见 www.w3.org/TR/css3-values/#calc。

当宽度值重新设定为 30%以后，为容器在.feature 规则里增加 box-sizing 属性，需要加上-moz-和-webkit-前缀版本：

```
.feature {
    float: left;
    -moz-box-sizing: border-box;
    -webkit-box-sizing: border-box;
    box-sizing: border-box;
    width: 30%;
    margin: 0 4.5% 0 0;
    padding: 130px 15px 5px 15px;
    border: 1px dashed #3C9;
    background-color: hsla(0,0%,100%,.3);
    background-repeat: no-repeat;
    background-position: top center;
}
```

-moz-版本是 Firefox 专用的， -webkit-版本同时影响 Safari 和 Chrome，而没有浏览器前缀的版本将会作用于以后的 Opera 和 IE 8 之后的版本。现在每个容器占用的总空间还是 30%，边框和内边距的占用空间包含在这之内，这样容器内的内容所占据的空间就减少了 32 像素。这种方式使得这三个容器在任何情况下都正好占满一行（图 7-29）。

表7-5 box-sizing属性的浏览器支持度

IE	Firefox	Opera	Safaril	Chrome
支持，8+	支持，需要-moz-前缀	支持	支持，需要-webkit-前缀	支持，需要-webkit-前缀

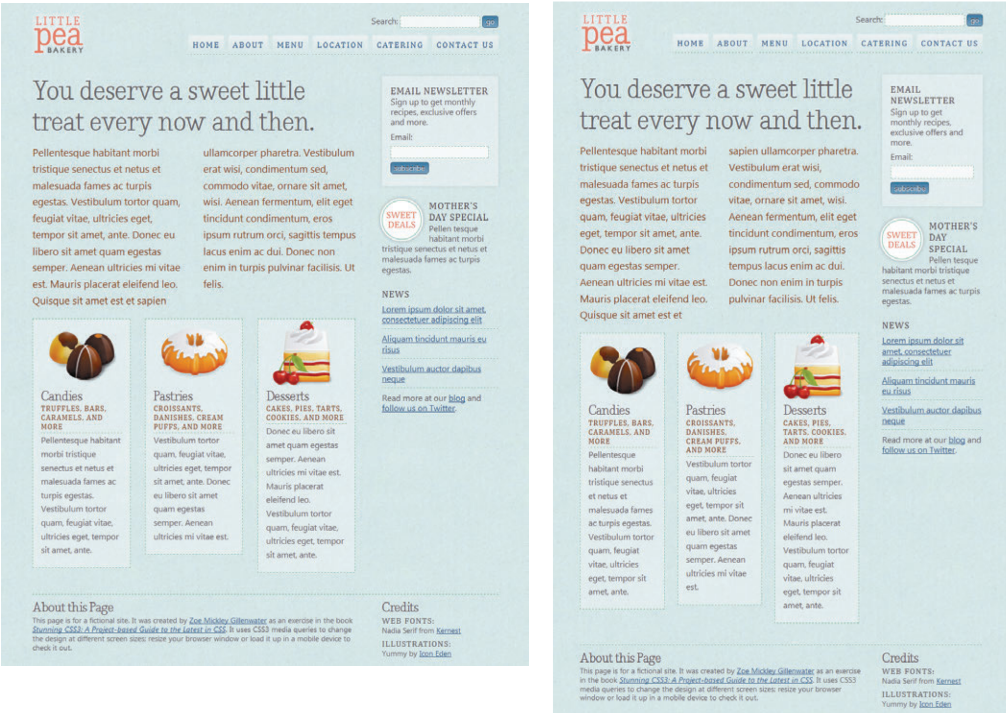


图 7-29　设置了 box-sizing 之后，盒模型变成了边框盒模型，三个容器重新排成一行

box-sizing 属性的来龙去脉

box-sizing 属性是基本用户界面（Basic User Interface）模块的一部分，你可以在 www.w3.org/tr/css3-ui 找到它。它能够让页面的盒模型在内容盒模型和边框盒模型之间转换。内容盒模型的边框和内边距的宽度是额外附加在容器之外的，而边框盒模型则会从容器内部切分出空间供边框和内边距使用。

Firefox 支持第三个值，padding-box，这个值意味着只有内边距的宽度是从容器内部切分出来的，边框的宽度还是额外附加到容器之外。这个值不是 W3C 的规范，W3C 貌似也没有添加它的打算。

box-sizing 属性可以用于如下一些场合。

- 在一个单一容器上使用混合的计量单位，这样就可以使用 box-sizing 来确认这个容器所占用的总的空间大小。
- 让一个容器 100%的拉伸适应父容器，但与此同时这个容器还能有边框和内边距。

IE 7 及之前版本的兼容方案

IE 7 及之前的版本并不支持 box-sizing。跟弹性布局模型类似，box-sizing 和纯粹装饰性效果的属性不同，所以必须为不支持的浏览器提供替代方案。在本节的例子中，最容易实现的

替代方案就是按照百分比而不是像素值来设置内边距，然后相应的减少容器的宽度百分比。添加如下的样式规则：

```
.ie6 .feature, .ie7 .feature {
    width: 25.5%;
    padding: 130px 2% 5px 2%;
}
```

这会有效地把每个容器的总宽度限制到 30%——两边各 2%的内边距加上 25.5%的宽度总共是 29.5%，这样是为了留下一点点空间给基于像素的边框。虽然按照百分比定义的内边距并不太理想，但是它看起来和基于像素定义的内边距的区别不是特别大，而且它至少会比第三个容器被迫换行的状况要强得多。

另外一个可选的替代方案是使 IE 进入怪异模式。在这个模式下 IE 会使用类似边框盒模型的定义来计算空间。但我并不建议这样做，首先，使 IE 7 进入怪异模式本身就是一件很难的事情。通常它会要求使用一个古老的或者非法格式的文档类型，这对其他浏览器会造成问题。其次，一旦进入怪异模式，那么整个网站的容器都将使用边框盒模型，这样之前工作正常的一些容器将被迫需要进行修改。

最后一种 box-sizing 的替代方案是使用脚本。Dean Edwards 的“IE 7”脚本（http://code.google. com/p/ie 7-js）能够让 box-sizing 在 IE 7 上工作。当然，目前只有这两个版本的 js 才能正常工作：IE 8.js 和 IE 9.js。使用脚本的缺点是脚本可能会被禁用，而 box-sizing 这个属性对于总的布局和外观来讲是比较重要的，所以依赖脚本解决问题会有一点小风险。

7.4.2　未来的布局系统

弹性布局模型并不是 CSS3 中唯一的新布局方式。还有两个布局系统，称为模板布局和网格布局。这两者都在开发阶段（特别是网格布局），目前还没有浏览器可以支持它们，甚至还没有浏览器计划这么做。但现在简单介绍一下还是可以让我们对未来布局系统的发展有所了解。

1. 模板布局

CSS3 模板布局模型（以前称为“高级布局”）能让开发者将内容放置到页面布局的“插槽”中去。通过将 display 的值设置成一个字母集合可以定义页面的“插槽”，就像 ASCII 码绘图艺术一样，举例来说，可以像下面这样定义：

```
body {
    display: "aaa"
             "bcc";
             "ddd";
}
```

然后可以将子容器放置到这些字母插槽里面去，就像下面所示：

```
#header { position: a; }
#content { position: b; }
#sidebar { position: c; }
#footer { position: d; }
```

当然这个系统可以比上面所示的复杂很多——我们还没有定义各个插槽的宽度，每一行的高度，或者元素的间距。但是重点在于，可以将一个 HTML 代码中的任意 div 放置到页面的任意位置，也可以轻易地创建多列布局。有些人会非常喜欢这里面的灵活性和精确性，也有些人可能会很讨厌 CSS3 模板布局背后的思想。

无论如何，在 www.w3.org/TR/css3-layout 可以了解关于模板布局的细节，如果想实际尝试一下的话，Alexis Deveria 的 jQuery 插件可以实现模板布局；参见 http://code.google.com/p/css-template-layout。

2. 网格布局

CSS3 网格布局甚至可以让你在不使用这些字母的情况下定义一个布局网格。只需要使用 `grid-columns` 和 `grid-rows` 这两个属性就能明确声明一个网格。浮动定位和绝对定位在这里还是需要的，但是这里的浮动和绝对都可以是相对于网格的，就是之前用新的 `gr` 单位定义的网格。跟新的 `float-offset` 属性一样，一个网格单位就相当于一栏的宽度。

网格布局的第一个草案（www.w3.org/TR/css3-grid）出现于 2007 年 9 月，一直到现在也没有任何更新。因此它的前景并不明朗。W3C 可能会考虑把模板布局和网格布局合并。即使它们还是独立发展，它们如何才能同时使用，以及如何和弹性布局模型一起使用，这些问题都非常不明晰。

附 录

浏览器支持度

下面的表格汇总了各个章节所涵盖的 CSS3 属性、选择器和一些特性的浏览器支持情况。少数原始表格下方提供有补充信息，它们在本附录里已被移除。

CSS3特性	所属章节	IE	Firefox	Opera	Safari	Chrome
`@font-face`	3	支持	支持，3.5+	支持，10+	支持	支持
`:nth-child()`	5	支持，9+	支持，3.5+	支持	支持	支持
`:nth-of-type()`	5	支持，9+	支持，3.5+	支持	支持	支持
`:target`	5	支持，9+	支持	部分支持	支持	支持
2D变形（2D Transforms）	2	不支持	支持，需要-moz-前缀，3.5+	支持，需要-o-前缀，10.5+	支持，需要-webkit-前缀	支持，需要-webkit-前缀
3D变形（3D Transforms）	2	不支持	不支持	不支持	支持，需要-webkit-前缀，5+	不支持
动画（Animation）	5	不支持	不支持	不支持	支持，需要-webkit-前缀	支持，需要-webkit-前缀
属性选择器（Attribute selectors）	4	支持，7+	支持	支持	支持	支持
`background-clip`	3	支持，9+	支持，4+；部分，1+，需要-moz-前缀	支持	支持，3+，需要 -webkit-前缀；部分支持，5+	支持
`background-size`		支持，9+	支持，4+；3.6+，需要-moz-前缀	支持	支持，5+；3+，需要-webkit-前缀	支持
`border-image`	3	不支持	部分支持，需要-moz-前缀，3.5+	部分支持，需要-o-前缀	部分支持，需要-webkit-前缀	部分支持
`border-radius`	2	支持，9+	支持，需要-moz-前缀	支持	支持，5+；4+，需要-webkit-前缀	支持

（续）

CSS3特性	所属章节	IE	Firefox	Opera	Safari	Chrome
box-shadow	2	部分支持，9+	支持，需要-moz-前缀，3.5+	支持，10.5+	支持，需要-webkit-前缀	支持，需要-webkit-前缀
box-sizing	7	支持，8+	支持，需要-moz-前缀	支持	支持，需要-webkit-前缀	支持，需要-webkit-前缀
弹性盒模型（Flexible Box Model）	7	不支持	部分支持，需要-moz-前缀	不支持	部分支持，需要-webkit-前缀	部分支持，需要-webkit-前缀
渐变（Gradients）	2	不支持	支持，3.6+，需要-moz-前缀	不支持	支持，需要-webkit-前缀	支持，需要-webkit-前缀
媒体查询（Media Queries）	6	部分支持，9+	部分支持，3.5+	部分支持	部分支持	部分支持
多栏（Multi-columns）	6	不支持	部分支持	不支持	部分支持	部分支持
多层背景图（Multiple Background Images）	3	支持，9+	支持，3.6+	支持，10.5+	支持	支持
RGBA/HSLA	2	支持，9+	支持	支持	支持	支持
text-shadow	2	不支持	支持	支持	支持	支持
转换（transition）	5	不支持	支持，需要-moz-前缀，4+	支持，需要-o-前缀，10.5+	支持，需要-webkit-前缀	支持，需要-webkit-前缀
word-wrap	2	支持，5.5+	支持，3.5+	支持	支持	支持

结束语

感谢你一路陪伴我历经这些吸引人的视觉效果、美妙的版式、强大的选择器，以及实用的可用性改进方案。你已经学会了如何通过使用这些高效、先进且合乎规范的 CSS 技巧为用户打造一个拥有尽可能优质和丰富体验的页面。

我希望你能将所学的技巧应用到开发中，无论是为了试验，或是打发时间，还是真的用来改进自己的作品。正如我先前所言，CSS3 仍旧在不断地发展之中，我们 Web 设计师也在持续地学习如何使它物尽其用。当你借助它的帮助实现了什么酷炫效果，请记得分享！可以通过 www.stunningcss3.com 上的表单写邮件给我，或是在 Twitter 上通过#stunningcss3 标签分享给整个 Web 设计社区。让我们从彼此的成果中汲取精华，共同使 Web 变得更加“令人惊艳”！

贡　　献

图表

图 1-1 www.w3.org

图 1-3～图 1-7 www.hsrc.unc.edu

图 1-8 和图 1-9 www.twitter.com

图 2-8 www.digitalmediaminute.com/reference/entity

图 2-26 www.useragentman.com/IETransformsTranslator

图 3-27 和图 3-28 www.fontsquirrel.com

图 5-8 http://en.wikipedia.org

图 7-23 www.cssstickyfooter.com

示例文件中的资源

第 3、4、5 章的示例文件中的水波纹图片是使用 Obsidian Dawn 所开发的 Photoshop 笔刷绘制而成（www.obsidiandawn.com/water-stains-photoshop-gimp-brushes）。

第 3、4、5 章的示例文件中所提供的 Prelude 字体是来自于 Font Squirrel（www.fontsquirrel.com）。

第 3、4、5 章的示例文件中所提供的图标是 Mark James 所设计的 famfamfam Silk 图标集的一部分（www.famfamfam.com/lab/icons/silk）。

第 5 章的示例文件中所提供的 Modernizr 脚本（modernizr-1.6.min.js）是由 Faruk Ateș 和 Paul Irish 所开发（www.modernizr.com）。

第 6 章的示例文件中所提供的 css3-mediaqueries-js 脚本（css3-mediaqueries.js）是由 Wouter van der Graaf 所开发（http://code.google.com/p/css3-mediaqueries-js）。

第 6、7 章的示例文件中所提供的图片是由 Icon Eden 所设计的 Yummy 图标集的一部分的（www.iconeden.com/icon/yummy-free-icons.html）。

第 6、7 章的示例文件中所提供的 Nadia Serif 字体来自于 Kernest（www.kernest.com）。

在此我对所有提供这些资源并准许我在书中使用它们的开发者以及设计师表示最衷心的感谢。

Stunning CSS3 A project-based guide to the latest in CSS

CSS3实用指南

CSS3为Web的视觉样式语言注入了强大的新功能，让设计人员更加轻松自如地设计优美而引人入胜的内容。借助CSS3，不使用图片就可以创建半透明背景、渐变、阴影等夺人眼球的视觉效果；还可以使用漂亮、独特、非Web安全的字体显示文本；不用Flash就可以创建动画；不用JavaScript就可以定制适应用户的设备和屏幕尺寸的设计。

本书通过一系列实用且新颖的范例，向读者展示如何实现以上功能和更多效果。每章都提供了几个实例，让你可以应用到正在开发的项目中，或从它们的设计思路中获取灵感。

本书还介绍了以下CSS3技术：

- 如何使用CSS3改善页面外观，提高可用性、可访问性和效率
- 处理不支持CSS3的早期浏览器的时机、解决方案和替代方案
- 如何利用独特的字体排版技术和漂亮的图片细节创建绝妙的设计
- 改进的新选择器，使标记更加流畅、不易出错
- 创建多列布局的新方法
- 如何不用脚本也能轻松制作出针对手机优化的Web设计

New Riders

图灵社区：www.ituring.com.cn
反馈/投稿/推荐信箱：contact@turingbook.com
热线：(010)51095186转604

分类建议 计算机/网页制作

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-27378-9

ISBN 978-7-115-27378-9
定价：49.00元